A Polynomial Sized Kernel for Tracking Paths Problem

Aritra Banik[1], Pratibha Choudhary[1], Daniel Lokshtanov[2], Venkatesh Raman[3], and Saket Saurabh[2,3]

[1] Indian Institute of Technology Jodhpur, India.
{aritrabanik|pratibhac247}@gmail.com
[2] University of Bergen, Bergen, Norway. {daniello|saket.saurabh}@ii.uib.no
[3] The Institute of Mathematical Sciences, HBNI, Chennai, India.
{vraman|saket}@imsc.res.in

Abstract. Consider a secure environment (say an airport) that has a unique entry and exit point with multiple paths between them. We want to place (minimum number of) trackers (or check points) at some specific intersections so that based on the sequence of trackers a person has encountered, we can identify the exact path traversed by the person. Motivated by such applications, we study the Tracking Paths problem in this paper. Given an undirected graph with a source s, a destination t and a non-negative integer k, the goal is to find a set of vertices of size at most k, a tracking set, that intersects each s-t path in a unique sequence. Such a set enables a central controller to *track* all the paths from s to t. We first show that the problem is NP-complete. Then we show that the finding a tracking set of size at most k is fixed-parameter tractable (FPT) when parameterized by the solution size. More specifically, given an undirected graph on n vertices and an integer k, we give a polynomial time algorithm that either determines that the graph can not be tracked by k trackers or produces an equivalent instance of size $\mathcal{O}(k^7)$.

1 Introduction

Tracking moving objects in a secure environment is an active area of research. Typically a secure environment is modeled as a network with fixed entry and exit point(s). Monitoring is achieved by placing small sensor nodes which monitor the movements of the objects in the network. For detailed study of field surveillance for the purpose of habitat monitoring, securing buildings, and intruder tracking please refer [3, 6]. While tracking traces of illegal activities over Internet, the biggest challenge is to track moving data packets [13, 15]. One may want to place trackers at an appropriate subset of routers in the network to track such activities.

Motivated by these applications, in a recent paper, Banik et. al. [2] considered the problem of target tracking theoretically and modeled it as the following graph theoretic problem. Let $G(V, E)$ be an undirected graph without any self loop or parallel edges and suppose G has a unique entry vertex s and a unique exit

vertex t. A simple path from s to t is called an s-t path. Let P be an s-t path in G and $A \subseteq V$ be a set of vertices. We denote by $\mathcal{T}_P^A$ the sequence of vertices of A obtained from P by deleting the vertices that do not belong to A. A set of vertices A is a tracking set for G, if and only if for any two distinct s-t paths P_1 and P_2, $\mathcal{T}_{P_1}^A \neq \mathcal{T}_{P_2}^A$. The vertices in set A are called trackers. Banik et.al. [2] proved that the problem of finding a minimum-cardinality tracking set with respect to *shortest* s-t paths (Tracking Shortest Paths problem) is NP-hard and APX-hard. In this paper we consider the problem of tracking all paths and not just shortest paths. In particular, we study the following.

Tracking Paths (G, s, t, k) **Parameter:** k
Input: An undirected graph $G(V, E)$ with two distinguished vertices s and t, and a non-negative integer k.
Question: Is there a tracking set T of size at most k for G?

Our Results and Methods. In this paper we study Tracking Paths from the perspective of parameterized complexity. Our first contribution is the following.

Theorem 1. Tracking Paths *is* NP*-complete.*

As is the case for any proof of NP-completeness, the proof of Theorem 1 requires two steps: hardness and containment inside NP. While hardness follows from the result of Banik et.al. [2] (See Appendix), it is not clear why the problem is in NP. To check whether a given set S is a tracking set for G, we need to go over all pairs of paths between s and t and check whether the sequence of trackers used on them are distinct. However, the number of paths between s and t could be exponential and thus it does not seem possible to exploit definition of the problem to show it inside NP. Thus, in order to show that the problem belongs to NP, we first give an alternate characterization for a set to be a tracking set. Then, to give a polynomial time algorithm to check whether a given set T is a tracking set, we first show that T is a *feedback vertex set* (FVS). That is, $G \setminus T$ is a forest. Using this property and our alternate characterization of tracking set we give a polynomial time algorithm to test whether a given set T is a tracking set.

Once we have shown that Tracking Paths is NP-complete, we study the problem from the viewpoint of parameterized complexity. In particular we design an FPT algorithm for Tracking Paths. That is, we design an algorithm for Tracking Paths with run time $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$, where k is the size of the tracking set we seek. In fact, we prove a stronger result than just designing an FPT algorithm; we give a polynomial kernel for the problem. In particular, given an instance (G, s, t, k), we give a polynomial time algorithm that either determines that (G, s, t, k) is a NO instance or produces an equivalent instance with $\mathcal{O}(k^6)$ vertices and $\mathcal{O}(k^7)$ edges. This polynomial time algorithm is called a *kernelization algorithm* and the reduced instance is called a *kernel*. For more details about parameterized complexity and kernelization we refer to monographs [4, 5]. Our second contribution is the following result.

Theorem 2. Tracking Paths *admits a polynomial kernel of size* $\mathcal{O}(k^7)$.

The kernelization algorithm (proof of Theorem 2) works along following lines. Let (G, s, t, k) be an input instance to Tracking Paths. We first apply a simple reduction rule to ensure that each edge and vertex in G belongs to some path from s to t. Then we prove two structural claims: (a) every tracking set S is an FVS; and (b) if G has a pair of vertices x and y such that x and y have at least $k+4$ vertex disjoint paths between them, then (G, s, t, k) is a NO instance. Next, using the known factor 2 approximation algorithm for the Feedback Vertex set problem, we compute a set S such that $G \setminus S$ is a forest. If $|S| > 2k$ then we immediately return that (G, s, t, k) is a NO instance. Thus, assume that S is of size at most $2k$. Now using the second structural claim regarding tracking set, we show that the number of connected components of $G \setminus S$ and the number of vertices in $V(G \setminus S)$ that have at least two neighbors in S are upper bounded by $k^{\mathcal{O}(1)}$. Next, we bound the number of vertices in $V(G \setminus S)$ that have *exactly* one neighbor in S. To do this, we fix a tree R in $G \setminus S$ and a vertex $v \in S$ and bound the size of the set of neighbors of v, $N_R(v)$, in R. Towards this, we consider the minimal subtree T in R that contains all the neighbors of v, and show that if $|N_R(v)| > k^{\mathcal{O}(1)}$, then we can partition the tree T into $k+1$ parts in such a way that each part must contain a tracker. This bounds the degree of each vertex in S into $V(G \setminus S)$ by $k^{\mathcal{O}(1)}$. This together with well-known counting methods on trees give us the desired polynomial kernel for Tracking Paths.

Related Work. Different structural properties of a graph have been studied previously to analyze navigational models in network setting. In a seminal paper Slater [14] introduced the concept of metric dimension of a graph. In graph theory, the metric dimension of a graph G is the minimum cardinality of a subset S of vertices such that all other vertices are uniquely determined by their distances to the vertices in S [7]. One application of metric dimension is the problem of determining the location of an object in a network depending on its distance from different landmarks in the network [11]. For a survey of metric dimension in graphs see [8]. Furthermore, as mentioned before Tracking Paths is also closely related to Feedback Vertex set in a graph. Feedback Vertex set is NP-hard [9], has a 2 approximation algorithm [1], a quadratic sized kernel [16] and an FPT algorithm running in time $\mathcal{O}((3.619)^k n^{\mathcal{O}(1)})$ [12].

Notations. A kernelization algorithm is obtained using what are called *reduction rules*. These rules transform the given parameterized instance in polynomial time to another equivalent instance, and a rule is said to be *safe* if the resulting graph has a tracking set of size at most k if and only if the original instance has one. For a path P, $V(P)$ denotes the vertex set of path P, and for a subgraph G', $V(G')$ denotes the vertex set of G'. Let P_1 and P_2 be two paths from v_a to v_b and from v_b to v_c respectively. By $P_1 \cdot P_2$, we denote the path from v_a to v_c created by concatenating P_1 and P_2.

2 NP-completeness

In this section we show that Tracking Paths is NP-complete. We first show that the problem is NP-hard. Towards that we will use the result of Banik et.al. [2]

which showed that Tracking Shortest Paths in a graph is NP hard by giving a reduction from Vertex Cover. We show that the same reduction also proves that the problem is NP hard when we want to track all s-t paths. In particular we prove the following hardness result in the appendix.

Lemma 1. $\circledast$[1] *For any graph $G(V, E)$ we can construct a graph G' on $|V|+|E|+5$ vertices such that, there exists a vertex cover of size k for G if and only if there exists a tracking set of size $k + |E| + 1$ for G'.*

In what follows, we show that the problem is NP-complete. Towards that we first give an alternate characterization for Tracking Paths. Then using a preprocessing rule, we show that every tracking set for a graph is also a feedback vertex set (FVS). Finally using these two properties we devise a polynomial time algorithm to check whether as given set of vertices is a tracking set for G.

2.1 Characterization of Tracking Set

Towards characterization of tracking set, we first define tracking set condition. For a graph $G(V, E)$ a set of vertices $V' \subseteq V$ is said to satisfy tracking set condition if there does not exist a pair of vertices $u, v \in V$, such that there exists two distinct paths, say P_1, P_2, between u and v in $G \setminus \{V' \cup \{s, t\}\} \cup \{u, v\}$, and, there exists a path from s to u, say P_{su}, and a path from v to t, say P_{vt} in $G \setminus \{V(P_1) \cup V(P_2)\} \cup \{u, v\}$, such that $V(P_{su}) \cap V(P_{vt}) = \emptyset$. Thus we have the following lemma.

Lemma 2. $\circledast$ *For a graph $G(V, E)$, a set of vertices $T \subseteq V$ is a tracking set if and only if T satisfies the* tracking set condition.

Although we have a nice characterization for what qualifies to be a tracking set, we still can not use it for verification purpose because there can be exponentially many paths between s and t. However, we show in the next subsection that in our case we can assume that between any two vertices we have only polynomially many relevant paths in the graph once we remove all the trackers along with s and t.

2.2 Tracking Set as Feedback Vertex Set

Let (G, s, t, k) be an input instance to Tracking Paths. After applying a reduction rule that ensures that each edge and vertex in G belongs to some s-t path, we can show that every tracking set is also an FVS for the reduced graph. For a graph $G(V, E)$, an FVS is a set of vertices $S \subseteq V$ such that $G \setminus S$ is a forest.

Reduction Rule 1 *If there exists a vertex or an edge that does not participate in any s-t path then delete it.*

[1] Proofs for lemmas/corollaries marked with $\circledast$ have been given in Appendix

Lemma 3. ⊛ *Reduction Rule 1 is safe and can be implemented in polynomial time.*

In the rest of the paper we assume that each vertex and each edge participates in at least one s-t path. Next we have a lemma which establishes the connection between a tracking set and FVS.

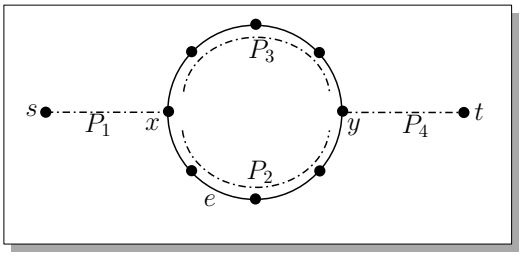

Fig. 1. Cycle without tracker

Lemma 4. *If T is a tracking set for G then T is a feedback vertex set for G as well.*

Proof. Consider any cycle C in G. We show that T contains at least one vertex from C. Consider an edge e in cycle C. Since every edge in the graph participates in at least one s-t path, let P be an s-t path that contains the edge e. Path P may contain some more vertices and edges from the cycle C. Let x be the first vertex in C present in P while traversing from s to t. Similarly let y be the last vertex of C in path P (see Figure 1). Observe that there are two paths from x to y in cycle C, one of them containing edge e, and the other one not containing edge e. Denote the path containing the edge e by P_2, and the other path by P_3. Let P_1 be the subpath (which would be empty if s is in the cycle) of P from s to x, and P_4 be the subpath (which would be empty if t is in the cycle) of P from y to t. Consider the following two paths:

$P' = P_1 \cdot P_2 \cdot P_4$

$P'' = P_1 \cdot P_3 \cdot P_4$

Observe that if C does not contain any tracker, then P' and P'' contain exactly the same sequence of trackers contradicting the fact that T is a tracking set. □

Thus we have the following corollary.

Corollary 1. *The size of a minimum tracking set T for G is at least the size of a minimum FVS for G.*

2.3 Verification of Tracking Set

In Lemma 4, we have shown that if T is a tracking set for G, then T is also an FVS for G. So we first check if T is an FVS for G. Using a breadth first search,

in $\mathcal{O}(n+m)$ time we can check whether $G \setminus T$ is a forest or not. If not, we reject T. Henceforth, we assume that T is an FVS for G.

Lemma 5. *For a set of vertices $T \subseteq V$, we can verify in polynomial time if T satisfies the* tracking set condition.

Proof. Observe that in order to show that $T \subseteq V$ is a tracking set it is sufficient to consider the subgraph G' of G, which only has those edges and vertices that are part of some s-t path (s). That is, we first apply Reduction Rule 1 and reduce given the instance. For the clarity of presentation we denote the reduced instance also by G.

Let $G' = G \setminus \{T \cup \{s,t\}\}$. In order to verify if tracking set condition holds for a set of vertices $T \subseteq V$, we first need to check if there exists a pair of vertices $u, v \in V$, such there there exists two distinct paths between u and v in $G' \cup \{u,v\}$. We consider each pair of vertices $u, v \in V$ and check in $G' \cup \{u,v\}$ if there exists two or more distinct paths between u and v. Observe that since a tracking set is also an FVS, G' is a forest. Hence there exists a unique path between each pair of vertices in G'. If both u and v belong to G', then there exists a unique path between them, and in such a case we can skip verification of second part of tracking set condition. If either u or v, or both of them do not belong to G', then they can have at most n neighbors each in G'. Hence, the number of paths between u and v in $G' \cup \{u,v\}$ is $\mathcal{O}(n^2)$. We consider each neighbor of u and v in $G' \cup \{u,v\}$, and find the unique path between these neighbors in G', in $\mathcal{O}(n)$ time using depth first search. Thus finding all paths between u and v can be done in $\mathcal{O}(n^3)$ time. For each pair of paths, say P_1 and P_2, among these $\mathcal{O}(n^2)$ paths, we verify second part of tracking set condition, i.e. we check if there exists a path from s to u, say P_{su}, and a path from v to t, say P_{vt} in $G \setminus \{V(P_1) \cup V(P_2)\} \cup \{u,v\}$, such that $V(P_{su}) \cap V(P_{vt}) = \emptyset$. This step can be performed in $\mathcal{O}(n^2)$ time using the algorithm for disjoint paths [10]. Hence the overall time taken for verification is $\mathcal{O}(n^2(n^3 + n^4 n^2))$. □

Lemmas 1 and 5 together prove Theorem 1.

3 Polynomial Kernel for Tracking Paths

In this section, with the help of some reduction rules we give a polynomial time algorithm that checks whether the given instance is a NO instance (for a solution of size at most k) or produces an equivalent instance with $\mathcal{O}(k^6)$ vertices. We assume that the given graph has been preprocessed using Reduction Rule 1.

Recall that from Corollary 1, we know that size of a minimum tracking set T for G is at least the size of a minimum FVS for G. Now we find a 2-approximate solution S in G for Feedback Vertex set using [1]. From Corollary 1, we have the following reduction rule.

Reduction Rule 2 *Apply the algorithm of* [1] *to find a 2-approximate solution, S for* Feedback Vertex set. *If $|S| > 2k$, then return that the given instance is a NO instance.*

Observe that $\mathcal{F} = G \setminus S$ is a forest. Now we try to bound the number of vertices in graph $\mathcal{F}$ given that k trackers are sufficient to track all the s-t path in G. Towards this we first prove a monotonicity lemma and a corollary which says that if a subgraph of G can not be tracked with k trackers, then G can not be tracked with k trackers.

Lemma 6. *Let T be a tracking set for $G(V, E)$ and $G'(V', E')$ be a subgraph of G such that $\{s, t\} \in V'$, and T' is a minimum tracking set for G'. Then $|T'| \leq |T|$.*

Proof. We show that T is a tracking set for G' as well. For otherwise, there must exist two s-t paths, say P_1 and P_2 in G' that contain the same sequence of trackers. Observe that P_1 and P_2 also belong to G. Hence, in this case P_1 and P_2 can not be distinguished by T in G as well. This contradicts the assumption that T is a tracking set for G. Hence the lemma holds. □

Corollary 2. *If a subgraph of G that contains both s and t can not be tracked by k trackers, then G can not be tracked by k trackers either.*

Henceforth, we limit ourselves to analyzing the cases when a subgraph can not be tracked by k trackers. First we bound the number of vertices in $\mathcal{F}$ that have at least two neighbors in S.

3.1 Bounding the number of vertices in $\mathcal{F}$ with at least two neighbors in S

Lemma 7. ⊛ *If there are two vertices $u, v \in V$ such that there exists more than $k + 3$ vertex disjoint paths between u and v, then G can not be tracked with k trackers.*

From Lemma 7 we have the following.

Reduction Rule 3 *If there exists two vertices in S that have $(k + 4)$ common neighbors in $\mathcal{F}$ then we return that the given instance is a NO instance.*

Lemma 8. ⊛ *If there exists two vertices in S that have $(k+4)$ common neighbors in $\mathcal{F}$ then G can not be tracked with k trackers.*

Corollary 3. ⊛ *If Reduction Rule 3 is not applicable, then the number of vertices in $\mathcal{F}$ that have at least two neighbors in S is at most $\binom{2k}{2}(k + 4) \leq 2k^2(k + 4)$.*

3.2 Bounding the number of trees in $\mathcal{F}$

The argument given in the proof of Lemma 8 can be also used to bound the number of trees in $\mathcal{F}$ that are adjacent to at least two vertices in S.

Reduction Rule 4 *If there exists two vertices in S that are common neighbors to $(k + 4)$ trees in $\mathcal{F}$, we return that the given instance is a NO instance.*

Lemma 9. ⊛ *If there are $(k+4)$ trees that are adjacent to two vertices u and v in S, then G can not be tracked with k trackers.*

Corollary 4. *If Reduction Rule 4 is not applicable, then the number of trees in $\mathcal{F}$ that have at least two neighbors in S is at most $\binom{2k}{2}(k+4)$.*

Next we bound the number of trees with exactly one neighbor in S. Towards this we first show the following.

Lemma 10. *Any induced subgraph G' of G containing at least one edge will contain a pair of vertices u, v such that there exists a path in G from s to u and a path from v to t, and these paths are mutually vertex disjoint and contain no other vertices from G' except u and v.*

Proof. Consider the induced subgraph G'. Pick any edge $e = (x, y)$ of G'. We know e participates in at least one s-t path, say P. Denote the subpath of P from s to x and from y to t by P_s and P_t respectively. Observe that P_s and P_t are vertex disjoint. Let $u \in G'$ be the first vertex in P_s while traversing from s to x and $v \in G'$ be the last vertex while traversing P_t from t to y. Observe that the subpath of P_s from s to u and the subpath of P_t from v to t are vertex disjoint and intersect with G' only at u and v. Therefore the claim holds. □

We show (in Appendix) that the only possible trees having exactly one neighbor in S are ones that contain s or t.

Lemma 11. ⊛ *The number of trees in $\mathcal{F}$ that have a single neighbor in S is at most two.*

From the above two lemmas we know that every tree has at least one neighbor in S. Hence we have the following corollary.

Corollary 5. *If the Reduction Rules 1 to 4 are not applicable, then the number of trees in $\mathcal{F}$ is at most $\binom{2k}{2}(k+4) + 2 \leq 2k^2(k+4)$.*

3.3 Bounding the number of vertices in $\mathcal{F}$ with exactly one neighbor in S

Here we bound the number of vertices in $\mathcal{F}$ that have a single neighbor in S and we do that by bounding the number of vertices from a single tree in $\mathcal{F}$ that are adjacent to a particular vertex of S.

Consider any tree $R \in \mathcal{F}$. Let V_f be the set of vertices of R that is adjacent to a particular vertex $f \in S$. In this section we prove a bound on the cardinality of V_f. We denote *the smallest connected subtree of R which contains all of V_f by R'*. Note that the leaf nodes of R' are in V_f. Consider three vertices $\{v_a, v_b, v_c\} \subset V_f$. Let R'' be any subtree of R' containing v_a, v_b and v_c. Given a path P and two vertices x, y on P by P_{xy} we denote the subpath of P starting at x and ending at y.

Lemma 12. *Any tracking set T must contain least one vertex of R''.*

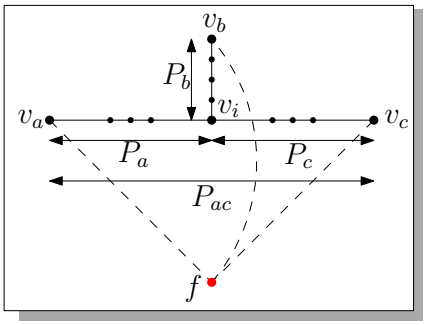

Fig. 2. Illustration of Lemma 12

Proof. Let T be a tracking set which does not contain any vertex from R''. Observe that all the leaf nodes of R'' are either v_a, v_b or v_c. Furthermore, at least two of v_a, v_b or v_c must be the leaves of R''. Without loss of generality assume that at least v_a and v_c are leaves of R''. Consider the path P_{ac} from v_a to v_c in R''. Without loss of generality assume that v_b is connected to P_{ac} via the path P_b (see Figure 2) joining P_{ac} at vertex v_i. We denote the paths from v_i to v_a and v_c by P_a and P_c respectively. Note that one of P_b could be a single vertex.

Let G' be the graph induced by $\{f\} \cup V(R'')$. From Lemma 10 we know that there exists a pair of vertices $u, v \in V(G')$, such that there are two vertex disjoint paths in G from s to u and from v to t and both these paths do not contain any vertex from G' except u and v. We use P_s to denote the path from s to u, and P_t to denote the path from v to t (see Figure 3(a)). Observe that depending on the locations of u and v there can be three cases.

Case 1 (u and v are in different chains among P_a, P_b and P_c): Without loss of generality assume $u \in P_a, v \in P_c$, other cases can be proved similarly. Denote the paths $P_{uv_a} \subset P_a$, $P_{v_iv} \subset P_c$ and $P_{v_cv} \subset P_c$ by λ_1, λ_2 and λ_3 respectively (see Figure 3(a)). Observe that $P_s \cdot \lambda_1 \cdot f \cdot P_b \cdot \lambda_2 \cdot P_t$ and $P_s \cdot \lambda_1 \cdot f \cdot \lambda_3 \cdot P_t$ contain the same sequence of trackers (note that f may or may not contain a tracker). Hence these two paths are indistinguishable which contradicts the fact that T is a tracking set.

Case 2 (u and v are in same chain among P_a, P_b and P_c): Without loss of generality assume $u, v \in P_a$. Denote the paths $P_{uv_a} \subset P_a$, $P_{v_cv} \subset P_{ac}$ and $P_{v_iv} \subset P_c$ by λ_1, λ_2 and λ_3 respectively (see Figure 3(b)). Observe that $P_s \cdot \lambda_1 \cdot f \cdot \lambda_2 \cdot P_t$ and $P_s \cdot \lambda_1 \cdot f \cdot P_b \cdot \lambda_3 \cdot P_t$ contain the same sequence of trackers. This contradicts that T is a tracking set.

Case 3 ($u = f$ or $v = f$): Without loss of generality assume $u = f$. Proof follows from the fact that from f there exists two paths to any other vertex in R''. In particular, $P_sP_{v_av}P_t$ and $P_sP_{v_bv}P_t$ are two paths that can not be distinguished.

This completes the proof. □

Next we show the following.

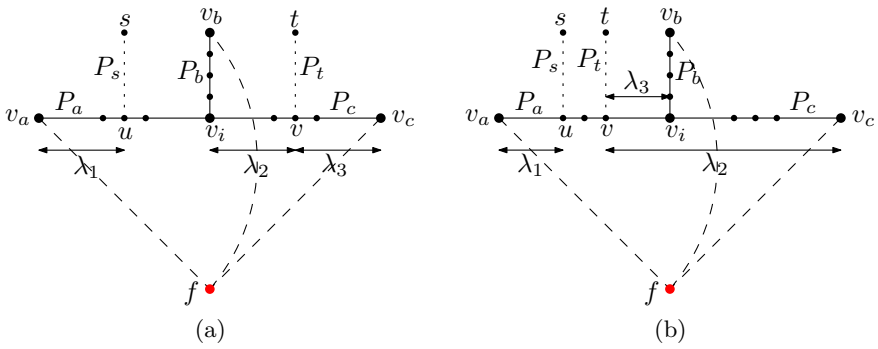

Fig. 3. Illustration of Lemma 12

Lemma 13. *The degree of each vertex in R' is at most $k+4$.*

Proof. Observe that if there exists a vertex v of degree $k+4$ in R', then there are at least $k+4$ disjoint paths between v and $k+4$ leaves of R' (see Figure 8 in Appendix). As every leaf node is connected with f, there will be more than $k+4$ disjoint paths between v and f. In that case from Lemma 7 we know that G can not be tracked with k trackers. Therefore we have a contradiction, and hence the result holds. □

Let the vertices in R' that are adjacent to f be colored RED and the others be colored BLUE. Since R' is a minimum tree containing V_f, we have that all the leaf nodes of R' are RED. Next we have following lemma.

Lemma 14. *If the number of* RED *vertices in R' is at least $(2k+8)(k+1)$ then we can partition R' into at least $k+1$ disjoint subtrees each containing at least three* RED *vertices.*

Proof. In R', let w be the closest node to a leaf such that the subtree rooted at w has at least three RED vertices. Since w is the closest node we have that for any of its children the subtree rooted at them has at most two RED children. Using Lemma 13, we can say this implies that the subtree rooted at w has at most $2(k+4)$ RED vertices. Now remove the subtree rooted at w, and continue. This implies that if $|V(R')| \geq (2k+8)(k+1)$, we can partition R' into at least $k+1$ disjoint subtrees each containing at least three RED vertices. □

From Lemma 12 we know that each such partition needs a tracker. Thus we have the following.

Lemma 15. *If there exists $(2k+8)(k+1)$ vertices of a tree in $\mathcal{F}$ adjacent to a single vertex in S, then G can not be tracked with k trackers.*

Reduction Rule 5 *If there is a vertex in S adjacent to $(2k+8)(k+1)$ vertices of a tree in $\mathcal{F}$, we return that the given instance is a NO instance.*

Corollary 6. ⊛ *If the Reduction Rule 5 is not applicable, then the total number of vertices from each tree in $\mathcal{F}$ that are adjacent to a vertex in S is at most $2k(2k+8)(k+1)$.*

3.4 Wrapping up – Polynomial kernel and FPT algorithm

From Corollaries 4 and 6, we have following corollary.

Corollary 7. *The number of vertices in a tree of $\mathcal{F}$ that have at least one neighbor in S is at most $2k^2(k+4) + 2k(2k+8)(k+1) = 2k(3k^2+14k+8)$.*

Next we give a reduction rule that helps bound the number of internal vertices in a tree in subsequent lemma.

Reduction Rule 6 *If there are three vertices v_a, v_b and v_c each of degree two, such that (v_a, v_b) and (v_b, v_c) both are edges, do the following. Delete v_b and introduce edge (v_a, v_c) in G.*

Lemma 16. ⊛ *Reduction Rule 6 is safe and can be implemented in polynomial time.*

Lemma 17. ⊛ *The number of vertices in a tree in $\mathcal{F}$ that do not have any neighbor in S is at most $10k(3k^2+14k+8)$.*

From Corollary 7 and Lemma 17, we have the following corollary.

Corollary 8. *The number of vertices in a tree in $\mathcal{F}$ is at most $12k(3k^2+14k+8)$ if none of the reduction rules are applicable.*

Proof of Theorem 2. From Corollaries 5 and 8 we can say that if none of the reduction rules are applicable, then the total number of vertices in $\mathcal{F}$ is at most $12k(3k^2+14k+8)2k^2(k+4)$ and hence the total number of vertices in G is at most $12k(3k^2+14k+8)2k^2(k+4)+2k$. Thus the total number of vertices is at most $72k^6+624k^5+1536k^4+768k^3+2k$ in G.

Since $\mathcal{F}$ is a forest, total number of edges in $\mathcal{F}$ is at most $12k(3k^2+14k+8)2k^2(k+4)-1$. From Reduction Rule 5 we know that total number of vertices from each tree in $\mathcal{F}$ that are adjacent to a single vertex in S is at most $2k(2k+8)(k+1)$. From Corollary 5 we know that number of trees in $\mathcal{F}$ is at most $2k^2(k+4)$. Thus the total number of edges between $\mathcal{F}$ and S is $2k(2k+8)(k+1)2k^2(k+4)2k$. Total number of edges among the vertices in S is at most $4k^2$. Thus the total number of edges in G is at most $12k(3k^2+14k+8)2k^2(k+4)-1+2k(2k+8)(k+1)2k^2(k+4)2k+4k^2 = \mathcal{O}(k^7)$. □

Theorem 3. Tracking Paths *is* FPT *when parameterized by the solution size, and run time of the* FPT *algorithm is $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$.*

Proof. We can run through all subsets of size k of the kernel and use the verification algorithm mentioned in Lemma 5 to check if a particular subset is a tracking set for G. Using the proof of Theorem 2, we can find all subsets of size k in $\mathcal{O}(k^{6k})$ time. From Lemma 5, we can verify if a subset of k vertices is a tracking set for G in polynomial time. Thus we have a $2^{\mathcal{O}(k \log k)} n^{\mathcal{O}(1)}$ time FPT algorithm for Tracking Paths. □

4 Conclusions

In this paper we have shown that the Tracking Paths problem is NP-complete. We also show the problem to be fixed-parameter tractable by proving the existence of a polynomial sized kernel. This is achieved via exploiting the connection between a feedback vertex set and tracking set. An open problem is to improve the size of the kernel and using it or otherwise to improve the run time of the FPT algorithm for the problem. Other directions to explore are to find approximation algorithms and studying the problem for directed graphs.

References

1. V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Discrete Math.*, 12(3):289–297, 1999.
2. A. Banik, M. J. Katz, E. Packer, and M. Simakov. Tracking paths. In *Algorithms and Complexity - 10th International Conference, CIAC 2017*, pages 67–79, 2017.
3. S. Bhatti and J. Xu. Survey of target tracking protocols using wireless sensor network. In *2009 Fifth International Conference on Wireless and Mobile Communications*, pages 110–115, 2009.
4. M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.
6. R. Gupta and S. R. Das. Tracking moving targets in a smart sensor network. In *IEEE 58th Vehicular Technology Conference*, volume 5, pages 3035–3039, 2003.
7. F. Harary and RA Melter. On the metric dimension of a graph. *Ars Combin*, 2(191–195):1, 1976.
8. C. Hernando, M. Mora, I. M. Pelayo, C. Seara, and D. R. Wood. Extremal graph theory for metric dimension and diameter. *The electronic journal of combinatorics*, 17(1):R30, 2010.
9. R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.
10. K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424 – 435, 2012.
11. S. Khuller, B. Raghavachari, and A. Rosenfeld. Landmarks in graphs. *Discrete Applied Mathematics*, 70(3):217 – 229, 1996.
12. T. Kociumaka and M. Pilipczuk. Faster deterministic feedback vertex set. *Information Processing Letters*, 114(10):556, 2014.
13. T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems. *ACM Comput. Surv.*, 39(1), 2007.
14. P. J Slater. Leaves of trees. *Congr. Numer*, 14(549–559):37, 1975.
15. A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. *SIGCOMM Comput. Commun. Rev.*, 31(4):3–14, 2001.
16. S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms (TALG)*, 6(2):32, 2010.

Appendix

Lemma 1. *For any graph $G(V, E)$ we can construct a graph G' on $|V| + |E| + 5$ vertices such that, there exists a vertex cover of size k for G if and only if there exists a tracking set of size $k + |E| + 1$ for G'.*

Proof. We start with the reduction due to Banik et. al. from Vertex Cover to the Tracking Shortest Paths. Let $\{G(V, E), k\}$ be an instance of the Vertex Cover for G. Construct the graph $G'(V', E')$ from $G(V, E)$ as follows. Create a vertex in G' for each vertex in G and for each edge in G. Also create vertices s and t, so that $V' = V \cup E \cup \{s, t\}$. In G' there is an edge between a vertex $v \in V$ and a vertex $e \in E$ if e is incident on v in G. For each vertex $v \in V$, create an edge between v and s in G', and for each vertex $e \in E$ create an edge between e and t in G' (see Figure 4). Add to V' the vertices a, b, d and to E' the edges $(s, a), (s, b), (d, t)$. Also add to E' the edges (a, x), for each $x \in E \cup \{d\}$, and the edge (b, d) (see Figure 4).

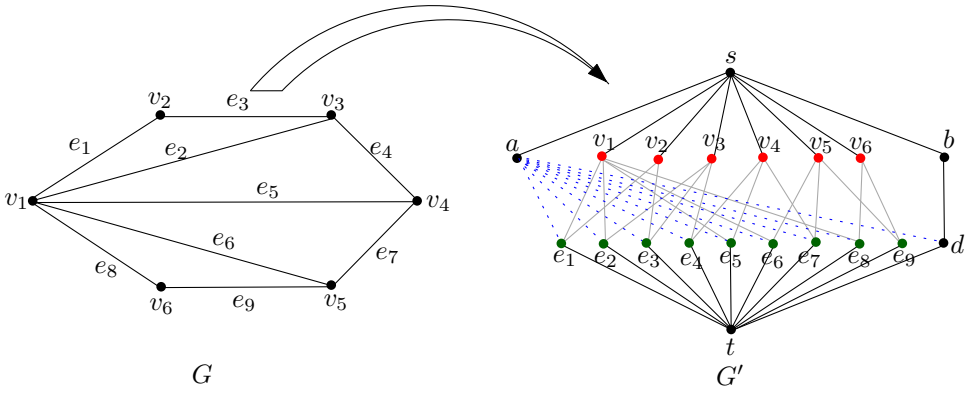

Fig. 4. Reduction from Vertex Cover to Tracking Paths

Observe that there exists a vertex cover in G of size k if and only if there exists a tracking set for shortest paths in G' of size $k + |E| + 1$.

We show that any tracking set for shortest paths in G' is also a tracking set for all s-t paths for G', thereby proving that there exists a vertex cover for graph G of size k if and only if there exists a tracking set for all s-t paths in G' of size $k + |E| + 1$.

Observation 1 *In G', a tracking set for shortest paths is also a tracking set for all s-t paths.*

Proof. Let T be any tracking set for shortest paths in G'. Assume that T is not a tracking set for all s-t paths in G'. Thus there exists two s-t paths (not necessarily shortest) P_1 and P_2 in G' that contain the same sequence of trackers.

Observe that each vertex in V' that represents an edge $e \in E$ is in T. Hence there exists two vertices $e_1, e_2 \in G'$ such that P_1 contains the sequence $e_1 \cdot v_i \cdot e_2$ and P_2 contains the sequence $e_1 \cdot v_j \cdot e_2$ where $v_i \neq v_j \in V$. Observe that both e_1 and e_2 are incident on v_i and v_j thus creating parallel edges and resulting in a contradiction. □

□

Lemma 2. *For a graph $G(V, E)$, a set of vertices $T \subseteq V$ is a tracking set if and only if T satisfies the* tracking set condition.

Proof. Let us assume that $T \subseteq V$ is a tracking set for G. We claim that T satisfies the tracking set condition. Suppose not. Then there exists a pair of vertices $u, v \in V$, such that there exists two distinct paths, say P_1, P_2, between u and v in $G \setminus \{T \cup \{s, t\}\} \cup \{u, v\}$, and, there exists a path from s to u, say P_{su}, and a path from v to t, say P_{vt} in $G \setminus \{V(P_1) \cup V(P_2)\} \cup \{u, v\}$, such that $V(P_{su}) \cap V(P_{vt}) = \emptyset$. Notice that there might be trackers in $V(P_{su}) \cup V(P_{vt})$, however there are no trackers in $V(P_1) \cup V(P_2) \setminus \{u, v\}$. Observe now there exists two s-t paths in G, $P_{su} \cdot P_1 \cdot P_{vt}$ and $P_{su} \cdot P_2 \cdot P_{vt}$ that have the same sequence of trackers. This contradicts the assumption that T is a tracking set for G.

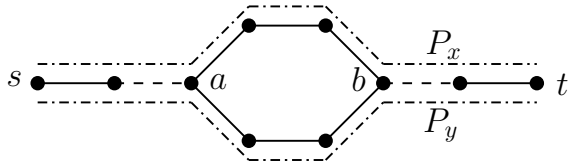

Fig. 5. Not satisfying tracking set condition

Conversely, let us assume that $T \subseteq V$ satisfies the tracking set condition. We claim that T is a tracking set for G. Suppose not. Then there exists at least two s-t paths in G, say P_x and P_y that contain the same sequence of trackers. Consider $V(P_x)$ and $V(P_y)$. Notice that s is the first vertex that appears in both P_x and P_y. Starting from s, keep scanning the vertices in P_x and P_y as long as the vertices in these paths are the same. Let a be the first vertex that appears in both P_x and P_y, such that after a the next vertex appearing in P_x is not the same as that appearing in P_y. Continue scanning the vertices in both paths, till a vertex b is found such that b appears in both P_x and P_y. See Figure 5. Observe that there exists two distinct paths between a and b in $G \setminus \{T \cup \{s, t\}\} \cup \{a, b\}$. Also, there must exist such a vertex a and a vertex b in $V(P_x) \cup V(P_y)$ such that there exist distinct paths between them, else P_x and P_y would not be two different s-t paths. Let us call these paths P_1 and P_2. Notice that $V(P_1) \cup V(P_2) \setminus \{a, b\}$ can not contain any tracker, else P_x and P_y would not contain the same sequence of trackers. There exists a path from s to a, say P_{sa}, and a path from b to t, say P_{bt} in $G \setminus \{V(P_1) \cup V(P_2)\} \cup \{a, b\}$, such that $V(P_{sa}) \cap V(P_{bt}) = \emptyset$, and both P_{sa} and P_{bt} may or may not contain any trackers. This contradicts the assumption that T satisfies the tracking set condition. □

Lemma 3. *Reduction Rule 1 is safe and can be implemented in polynomial time.*

Proof. Reduction Rule 1 is safe as a tracking set for the resulting graph will also be a tracking set for the original graph and vice versa. An edge $e = (u, v)$ participates in an s-t path if and only if there exists two simple paths, one from s to u and another from v to t, such that both these paths are mutually vertex disjoint (or simple paths from s to v and from u to t that are mutually vertex disjoint). Existence of such paths can be determined in quadratic time [10]. Note that if there exists degree one vertices in G that did not participate in any s-t path, removing all edges that do not participate in any s-t path will isolate these vertices. Now the vertices that do not participate in any s-t path will be only isolated vertices. In linear time we can remove such vertices from the graph. □

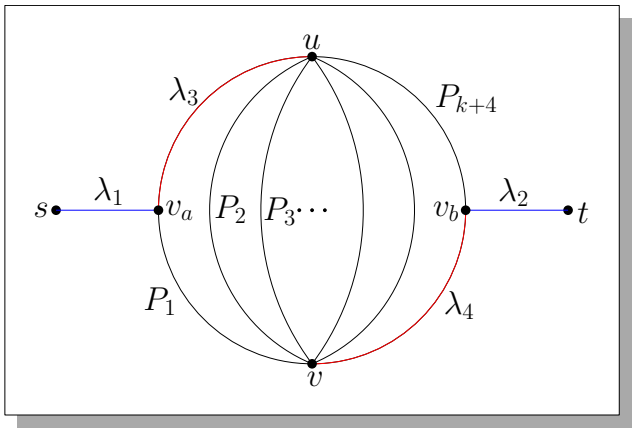

Fig. 6. Illustration of Lemma 7

Lemma 7. *If there are two vertices $u, v \in V$ such that there exists more than $k + 3$ vertex disjoint paths between u and v, then G can not be tracked with k trackers.*

Proof. For a contradiction assume that there exists $k + 4$ vertex disjoint paths $\mathcal{P} = \{P_1, \ldots, P_{k+4}\}$ between u and v, and G can be tracked with k trackers. Let G' be the subgraph induced by $\mathcal{P}$. As each edge and vertex of G is part of some s-t path, there is at least one s-t path, λ that intersects G'. Let v_a and v_b be the first and last vertex respectively visited by λ in G' (see Figure 6). We denote the subpaths from s to v_a and v_b to t by λ_1 and λ_2 respectively. Here it is possible that $V(\lambda_1) = \{s\} = \{v_a\}$ and/or $V(\lambda_2) = \{t\} = \{v_b\}$, but this does not change the number of vertex disjoint paths between u and v, and the proof still holds. Observe that either there exists a path from v_a to u that is vertex disjoint from the path from v to v_b, or there exists a path from v_a to v that is vertex disjoint from the path from u to v_b, and either pair of the paths can intersects with at

most two paths from $\mathcal{P}$. Without loss of generality, assume that there exists mutually disjoint paths between v_a to u, $\lambda_3 \subseteq P_1$ and v to v_b, $\lambda_4 \subseteq P_{k+4}$. By pigeonhole principle among $P_2 \setminus \{u, v\}, \ldots, P_{k+3} \setminus \{u, v\}$ there exists two paths that do not contain any trackers. Without loss of generality assume $P_2 \setminus \{u, v\}$ and $P_3 \setminus \{u, v\}$ do not contain any tracker. Consider following two paths.

$\lambda' = \lambda_1 \cdot \lambda_3 \cdot P_2 \cdot \lambda_4 \cdot \lambda_2$

$\lambda'' = \lambda_1 \cdot \lambda_3 \cdot P_3 \cdot \lambda_4 \cdot \lambda_2$

Observe λ' and λ'' are two s-t paths with the same sequence of trackers and this contradicts our assumption that G can be tracked by at most k trackers. Hence the result holds.

Note that if $\lambda_3 = P_1$ and $\lambda_4 = P_{k+4}$ i.e. $v_a = v$ and $v_b = u$, then the number of vertex disjoint paths between u and v should be $k+1$ or less. Also, if v_a and v_b lie on the same path in $\mathcal{P}$, then the number of vertex disjoint paths between u and v should be $k+2$ or less. Correctness of the proof follows from Corollary 2. □

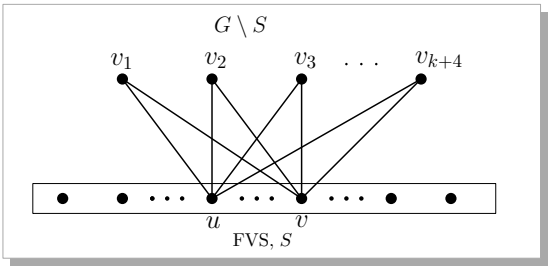

Fig. 7. $k+4$ vertices in $\mathcal{F}$ connected to same pair of vertices in S

Lemma 8. *If there exists two vertices in S that have $(k+4)$ common neighbors in $\mathcal{F}$ then there does not exist a tracking set of size at most k.*

Proof. Suppose there exists $k+4$ vertices in $\mathcal{F}$ that are neighbors of two distinct vertices u and v of S (see Figure 7). This leads to the formation of more than $k+4$ vertex disjoint paths between u and v, as shown in Figure 7. We know from Lemma 7 that in such a case G can not be tracked with k trackers. Correctness of the proof follows from Corollary 2. □

Corollary 3. *If Reduction Rule 3 is not applicable, then the number of vertices in $\mathcal{F}$ that have at least two neighbors in S is at most $\binom{2k}{2}(k+4) \leq 2k^2(k+4)$.*

Proof. For each such vertex, associate a pair of vertex in S to which it is adjacent. If there are more than $\binom{2k}{2}(k+4)$ such vertices then one pair of vertices in S will be associated to more than $k+4$ vertices of $\mathcal{F}$ and that pair of vertices in S will have at least $k+4$ common neighbors and Reduction Rule 3 will become applicable. □

Lemma 9. *If there are $(k+4)$ trees that are adjacent to two vertices u and v in S, then G can not be tracked with k trackers.*

Proof. Note that there is a unique path between any two vertices in a tree. The rest of the argument is the same as the argument used in Lemma 8. Correctness of the proof follows from Corollary 2. □

Lemma 11. *The number of trees in $\mathcal{F}$ that have a single neighbor in S is at most two.*

Proof. Let R be a tree in $\mathcal{F}$ that does not contain s or t. If R contains only one vertex, then it will have at least two neighbors in S, as due to Reduction Rule 1, G has no vertex with degree at most one or multiple edges. Hence R contains at least two vertices and hence an edge. By Lemma 10, there exists a pair of vertices u, v in R such that there is a path in G from s to u and a path from v to t that are mutually disjoint and contain no other vertex of R. This means that R has at least two neighbors outside R (the neighbors of u and v in those paths) and hence in S.

So the only possible trees having a single neighbor in S are those that contain s or t, and hence the lemma follows. □

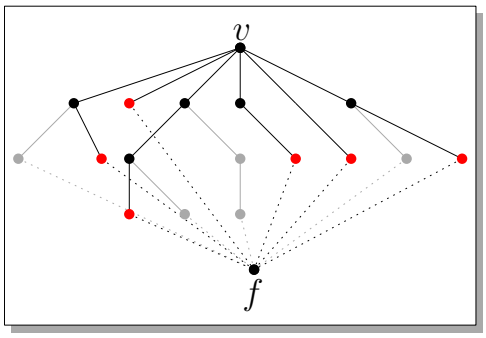

Fig. 8. Degree of each vertex in R' is at most $k + 4$

Corollary 6. *If the Reduction Rule 5 is not applicable, then the total number of vertices from each tree in $\mathcal{F}$ that are adjacent to a vertex in S is at most $2k(2k + 8)(k + 1)$.*

Proof. Otherwise, there will be a vertex in S that will have more than $(2k + 8)(k + 1)$ nodes in a tree and Reduction Rule 5 will apply. □

Lemma 16. *Reduction Rule 6 is safe and can be implemented in polynomial time.*

Proof. If there exists an edge between v_a and v_c, v_a, v_b, v_c would form an isolated triangle whose edges would not participate in any s-t path. Since Reduction Rule 1 removes all such edges from graph, such a case is not possible. Now observe that since by removing v_b the graph structure is not changed, except that the s-t paths passing through v_b have their lengths reduced by one vertex, proving

that there exists an optimal tracking set that does not contain v_b suffices. Let T be any optimal tracking set which contains v_b. Observe that any s-t path that contains either of the three vertices, say v_b, must also contain the other two vertices v_a and v_c. Thus a single vertex (any one) amongst v_a, v_b, v_c is sufficient to be included in T to indicate containment of vertices in an s-t path if required. Two vertices amongst v_a, v_b, v_c have to be included in T when their mutual sequence is different in two s-t paths with the same vertex sets.

First consider the case when T does not contain v_a and v_c, but contains v_b. Observe that in any s-t path P that contains v_b, the relative ordering between v_b and some other vertex v_d in that path P is same as the relative ordering between v_c and v_d. Hence $T \setminus \{v_b\} \cup \{v_c\}$ is also a tracking set.

Next consider the case when T contains v_a and v_b. Since relative ordering of v_a and v_b is same as that of v_a and v_c, we can replace v_b by v_c in this case. Thus $T \setminus \{v_b\}$ is also a tracking set. Hence the reduction rule is safe. This rule can be implemented in polynomial time using a standard depth first search. □

Lemma 17. *The number of vertices in a tree in $\mathcal{F}$ that do not have any neighbor in S is at most $10k(3k^2 + 14k + 8)$.*

Proof. In a tree, we denote the set of leaf nodes by V_1, the set of vertices of degree two by V_2, and the set of vertices of degree three or more by V_3. Since G is preprocessed, there can not be any degree one vertex in G except s and t. Thus each leaf of the tree in $\mathcal{F}$ necessarily has a neighbor in S. Hence by Corollary 7, number of vertices in V_1 is bounded by $2k(3k^2 + 14k + 8)$. Observe that both V_2 and V_3 can belong to the set of internal nodes in a tree. By standard graph theoretic properties of a tree, we know that $|V_3| \leq |V_1| - 1$. Hence the number of vertices in V_3 is bounded by $2k(3k^2 + 14k + 8) - 1$. Due to Reduction Rule 6, we know that at most two vertices of degree two can exist in series. Observe that each vertex in V_1 and V_3 can have at most two vertices from V_2 as its immediate ancestors. Hence the number of vertices in V_2 is bound by $2(|V_1| + |V_3|)$, i.e. $|V_2| \leq 8k(3k^2 + 14k + 8) - 2$. Hence $|V_2| + |V_3| \leq 2k(3k^2 + 14k + 8) - 3$. We ignore this reduction in count by 3 from our bound. Thus the overall bound on number of vertices in $\mathcal{F}$ which do not have any neighbor in S is $10k(3k^2 + 14k + 8)$. □