Polynomial Time Algorithms for Tracking Path Problems

Pratibha Choudhary(✉)

Indian Institute of Technology Jodhpur, Jodhpur, India
pratibhac247@gmail.com

Abstract. Given a graph G, and terminal vertices s and t, the TRACKING PATHS problem asks to compute a minimum number of vertices to be marked as trackers, such that the sequence of trackers encountered in each s-t path is unique. TRACKING PATHS is NP-hard in both directed and undirected graphs in general. In this paper we give a collection of polynomial time algorithms for some restricted versions of TRACKING PATHS. We prove that TRACKING PATHS is polynomial time solvable for chordal graphs and tournament graphs. We prove that TRACKING PATHS is NP-hard in graphs with bounded maximum degree $\delta \geq 6$, and give a $2(\delta + 1)$-approximate algorithm for the same. We also analyze the version of tracking s-t paths where paths are tracked using edges instead of vertices, and we give a polynomial time algorithm for the same. Finally we give a polynomial algorithm which, given an undirected graph G, a tracking set $T \subseteq V(G)$, and a sequence of trackers π, returns the unique s-t path in G that corresponds to π, if one exists.

Keywords: Graphs · Paths · Chordal graphs · Tournaments · Approximation · Bounded degree graphs · Tracking paths

1 Introduction

Tracking moving objects in networks has been studied extensively due to applications in surveillance and monitoring. Specific cases include secure system surveillance, habitat monitoring, vehicle tracking, and other similar scenarios. Object tracking in networks also finds applications in analyzing disease spreading patterns, information dissemination patterns on social media, and data packet flow in large networks like the world wide web. Tracking has been largely studied in the fields of machine learning, artificial intelligence, networking systems among other fields.

The problem of tracking paths in a network was first graphically modeled by Banik et al. in [2]. Let $G = (V, E)$ be an undirected graph without any self loops or parallel edges and suppose that G has a unique entry vertex (source) s and a unique exit vertex (destination) t. A simple path from s to t is called

This work was done while the author was visiting The Institute of Mathematical Sciences, Chennai, India.

an s-t path. The problem requires finding a set of vertices $T \subseteq V$, such that for any two distinct s-t paths, say P_1 and P_2, in G, the sequence of vertices in $T \cap V(P_1)$ as encountered in P_1 is different from the sequence of vertices in $T \cap V(P_2)$ as encountered in P_2. Here T is called a *tracking set* for the graph G, and the vertices in T are referred to as *trackers*. Banik et al. [2] proved that the problem of finding a minimum-cardinality tracking set to track *shortest* s-t paths (Tracking Shortest Paths problem) is NP-hard and APX-hard. Later, the problem of tracking all s-t paths (Tracking Paths) in an undirected graph was studied in [4,7,10]. Tracking Paths is formally defined as follows.

Tracking Paths (G, s, t)
Input: An undirected graph $G = (V, E)$ with terminal vertices s and t.
Question: Find a minimum cardinality tracking set T for G.

Tracking Paths was proven to be NP-complete in [4]. Here, the authors studied the parameterized version of Tracking Paths, which asks if there exists a tracking set of size at most k, and showed it to be fixed-parameter tractable (FPT) by giving a polynomial kernel. Specifically, it was proven that an instance of Tracking Paths can be reduced to an equivalent instance of size $\mathcal{O}(k^7)$ in polynomial time, where k is the desired size of the tracking set. In [7], the authors improved this kernel to $\mathcal{O}(k^2)$, and gave an $\mathcal{O}(k)$ kernel for planar graphs. In [10], Eppstein et al. proved that Tracking Paths is NP-complete for planar graphs and gave a 4-approximation algorithm for this setting. Here, the authors also proved that Tracking Paths can be solved in linear time for graphs of bounded clique width, when the clique decomposition is given in advance.

Tracking Shortest Paths was also studied in [3] and [5]. In [3], Banik et al. studied Tracking Shortest Paths and proved the problem to be fixed-parameter tractable. In [5], Bilò et al. prove that Tracking Shortest Paths is NP-hard for cubic planar graphs in case of multiple source-destination pairs, and give an FPT algorithm parameterized by the number of vertices equidistant from the source s.

In this paper we study Tracking Paths for chordal graphs, tournament graphs, and degree bounded graphs. A *chordal* graph is a graph in which each cycle of length greater than three has a chord (an edge between non-adjacent vertices of the cycle). A *tournament* is a directed graph in which there exists a directed edge between each pair of vertices. So far all the work done on Tracking Paths has been focused on tracking s-t paths (or shortest s-t paths) using vertices. In this paper, we also study tracking s-t paths using edges. We also give a path reconstruction algorithm that finds the unique s-t path corresponding to the given sequence of trackers, if one exists. Chordal graphs find applications in computational biology, computer vision and artificial intelligence [9,13,16,18]. Tournament graphs are used in voting theory and social choice theory to graphically depict pairwise relationships between entities in a community [17,19]. Tournament graphs are particularly used to study the Condorcet voting model, where a preference is indicated between each pair of contestants [11].

Our Results and Methods. In this paper we give polynomial time results for some variants of the Tracking Paths problem. We prove that Tracking Paths is polynomial time solvable for chordal graphs and tournaments. The key idea in proofs for chordal and tournament graphs is that if two s-t paths differ in only one vertex, than that vertex necessarily needs to be marked as a tracker. Next we prove that Tracking Paths is NP-hard for graphs with maximum degree δ ($\delta \geq 6$). We also give a $2(\delta + 1)$-approximation algorithm for graphs with maximum degree δ. Here the idea is to ensure that sufficient vertices are marked as trackers in each cycle. This derives from the fact that each cycle in a graph necessarily needs a tracker [4]. In order to give a complete solution for tracking paths in a graph, we also give an algorithm that reconstructs the required s-t path given a sequence of trackers and a constant size tracking set for the input graph. This uses the fact that by the definition of a tracking set, each maximal sequence of trackers in a tracking set should correspond to at most one s-t path in a graph. The reconstruction algorithm uses the disjoint path algorithms for undirected graphs [15] and tournaments [8] to construct the required s-t path.

Towards the end of the paper we analyze the problem of tracking s-t paths in an undirected graph using edges rather than vertices. We prove that even while using edges, each cycle in the graph needs at least one edge to be marked as a tracker. Further, a minimum feedback edge set (set of edges whose removal makes a graph acyclic) is also a minimum tracking edge set.

2 Notations and Definitions

Throughout the paper, while analyzing tracking paths using vertices in a graph, we assume graphs to be simple i.e. there are no self loops and multi-edges. When considering tracking set for a graph $G = (V, E)$, we assume that the given graph is an s-t graph, i.e. the graph contains a unique source $s \in V$ and a unique destination $t \in V$ (both s and t are known), and we aim to find a tracking set that can distinguish between all simple paths between s and t. Here s and t are also referred as the terminal vertices. If $a, b \in V$, then unless otherwise stated, $\{a, b\}$ represents the set of vertices a and b, and (a, b) represents an edge between a and b. For a vertex $v \in V$, *neighborhood* of v is denoted by $N(v) = \{x \mid (x, v) \in E\}$. We use $deg(v) = |N(v)|$ to denote degree of vertex v. For a subgraph G', $V(G')$ represents the vertex set of G' and $E(G')$ represents those edges whose both endpoints belong to $V(G')$. For a vertex $v \in V$ and a subgraph G', $N_{G'}(v) = N(v) \cap V(G')$. For a subset of vertices $V' \subseteq V$ we use $N(V')$ to denote $\bigcup_{v \in V'} N(v)$. With slight abuse of notation we use $N(G')$ to denote $N(V(G'))$. For a graph G and a set of vertices $S \subseteq V(G)$, $G - S$ denotes the subgraph induced by the vertex set $V(G) \setminus V(S)$. If S is a singleton, we may use $G - x$ to denote $G - S$, where $S = \{x\}$. A *chord* in a cycle is an edge between two vertices of the cycle, such that the edge itself not part of the cycle.

In an undirected graph, a feedback vertex set (FVS) is a set of vertices whose removal makes the graph acyclic and feedback edge set (FES) is the set of edges whose removal makes the graph acyclic. An edge-weighted graph is a graph with real valued weights assigned to each of its edges. Let P_1 be a path between vertices a and b, and P_2 be a path between vertices b and c, such that $V(P_1) \cap V(P_2) = \{b\}$. By $P_1 \cdot P_2$, we denote the path between a and c, formed by concatenating paths P_1 and P_2 at b. Two paths P_1 and P_2 are said to be *vertex disjoint* if their vertex sets do not intersect except possibly at the end points, i.e. $V(P_1) \cap V(P_2) \subseteq \{a, b\}$, where a and b are the starting and end points of the paths. By distance we mean length of the shortest path, i.e. the number of edges in that path. For a sequence of vertices π, by $V(\pi)$ we mean the set of vertices in the sequence π. If there exists a path P such that (a, b) is an edge that lies at one end point of P, then $P - (a, b)$ denotes the subpath of P obtained after removing the edge (a, b). Graphs which have maximum degree of vertices as three are known as *cubic* graphs. By a *bounded degree graph*, we mean a graph whose vertices have a maximum degree of d, where d is some constant.

3 Preliminary Analysis

In this section, we give some basic claims which are used for proving results in subsequent sections. We start by first recalling a reduction rule from [4] that ensures that each vertex and edge in the input graph participates in an s-t path.

Reduction Rule 1. *[4] In a graph G, if there exists a vertex or an edge that does not participate in any s-t path in G, then delete it.*

It is known that Reduction Rule 1 is safe and can be applied in quadratic time on undirected graphs [4]. In the rest of the paper, by *reduced graph* we mean a graph that is preprocessed using Reduction Rule 1. Let G' be a subgraph of graph G, and $u, v \in V(G')$. If there exists a path in G from s to u, say P_{su}, and another path from v to t, say P_{vt}, such that $V(P_{su}) \cap V(P_{vt}) = \emptyset$, $V(P_{su}) \cap V(G') = \{u\}$ and $V(P_{vt}) \cap V(G') = \{v\}$, then u is a *local source* for G' and v is a *local destination* for G'. Next we recall the following lemma from [4], which is used to define some commonly used terms in this paper.

Lemma 1. *In a reduced graph G, any subgraph G' consisting of at least one edge, contains a local source and local destination.*

Now we state the *tracking set condition*, which is useful for validation of a tracking set [4].

Tracking Set Condition:
For a graph $G = (V, E)$, with terminal vertices $s, t \in V$, a set of vertices $T \subseteq V$, is said to satisfy the tracking set condition if there does not exist a pair of vertices $u, v \in V$, such that the following holds:

- there exist two distinct paths, say P_1 and 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 $(G \setminus (V(P_1) \cup V(P_2))) \cup \{u, v\}$, and $V(P_{su}) \cap V(P_{vt}) = \emptyset$, i.e. P_{su} and P_{vt} are mutually vertex disjoint, and also vertex disjoint from P_1 and P_2.

It is known that for a reduced graph G, a set of vertices $T \subseteq V(G)$ is a tracking set if and only if T satisfies the *tracking set condition*. We use this fact, to prove the following lemma.

Lemma 2. ⊛[1] *In a graph G, if $T \subseteq V(G)$ is not a tracking set for G, then there exist two s-t paths with the same sequence of trackers, and they form a cycle C in G, such that C has a local source a and a local destination b, and $T \cap (V(C) \setminus \{a, b\}) = \emptyset$.*

4 Tracking Paths in Chordal Graphs and Tournaments

In this section, we consider polynomial time algorithms for solving Tracking Paths for chordal graphs and tournaments.

We start by giving a polynomial time algorithm for finding a tracking set for undirected chordal graphs. Recall that chordal graphs are those graphs in which each cycle of length greater than three has a chord. Many problems that are known to be NP-hard on general graphs are polynomial time solvable for chordal graphs e.g. chromatic number, feedback vertex set, independent set [14].

In undirected graphs, a tracking set is also a feedback vertex set [4]. However, a tracking set can be arbitrarily larger in size compared to a feedback vertex set. This holds true for chordal graphs as well.

Algorithm 1: Finding Tracking Set for a Chordal Graph.

```
Input: Chordal graph $G = (V, E)$ and vertices $s, t \in V$.
Output: Tracking Set $T \subseteq V$ for $G$.
1 Initialize $T = \emptyset$; Apply Reduction Rule 1;
2 foreach $e = (a, b) \in E$ do
3     foreach $x \in (N(a) \cap N(b)) \setminus T$ do
4         if $\exists$ an $s$-$t$ path $P$ in $G - x$ such that $e \in E(P)$ then
5             $T = T \cup \{x\}$;
6         end
7     end
8 end
9 Return $T$;
```

Algorithm 1 gives a procedure to compute a minimum tracking set for a chordal graph G. We prove its correctness in the following lemma.

Lemma 3. *Algorithm 1 gives an optimum tracking set for a chordal graph.*

[1] Proofs of Lemmas marked with ⊛ can be found in the full version of the paper [6].

Proof. Algorithm 1 first ensures that each vertex and edge in the input graph G participates in an s-t path. Next for each edge $e = (a, b) \in E$, if there exists a vertex $x \in (N(a) \cap N(b)) \setminus T$, we check if there exists an s-t path in $G - x$ that contains the edge e. Let P such a path in $G - x$. Now consider the path P' that can be obtained by replacing the edge e in P by the path $(a, x) \cdot (b, x)$ along with the vertex x. Observe that the vertex sets of P and P' differ only in vertex x. Hence, x necessarily belongs to a tracking set for G.

Now we prove that Algorithm 1 indeed returns an optimal tracking set T for G. Suppose not. Then T is not a tracking set for G. Due to Lemma 2, there exists two s-t paths, say P_1, P_2 and they form a cycle C in G, such that C has a local source u and a local destination v, and $V(C) \setminus \{u, v\}$ does not contain any trackers. See Fig. 1.

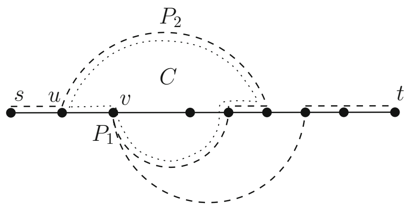

Fig. 1. Indistinguishable s-t paths in a graph form a cycle (marked in dotted lines)

Path P_1 is marked in solid lines, while path P_2 is marked in dashed lines. Since P_1 and P_2 contain the same sequence of trackers, no vertex in $V(C) \setminus \{u, v\}$ can be a tracker. Since we consider graphs without any parallel edges, there exists at least one vertex in $V(C) \setminus \{u, v\}$.

First, consider the case where C is a triangle. Due to Algorithm 1, the vertex in $V(C) \setminus \{u, v\}$ would have been marked as a tracker. This contradicts the assumption that no vertex in $V(C) \setminus \{u, v\}$ is marked as a tracker.

Next, consider the case when C is not a triangle (C contains four or more vertices). Since G is a chordal graph, C contains a chord. Consider the following two cases based on whether a chord is incident on the vertex u or not. Let w and x be two vertices such that $w, x \in N(u) \cap V(C)$ and $(w, x) \in E$. Without loss of generality, let $x \in V(P_1)$. Observe that the edge (u, x) in path P_1 can be replaced by the concatenated path $(u, w) \cdot (w, x)$, to obtain a new path that differs from P_1 only at the vertex w. Hence w must have been marked as a tracker by Algorithm 1. Next, consider the case when a chord in C is incident on u. Let $a \in N(u) \cap V(C)$ and $b \in N(a) \cap V(C)$, such that $a \neq b \neq u$, and $(b, u) \in E$. Note that there exists an s-t path containing the edge (b, u), where (b, u) can be replaced with the path $(b, a) \cdot (a, u)$, to obtain a new path that differs only at the vertex a. Hence a must have been marked as a tracker by Algorithm 1. Both the above cases contradict the assumption that no vertex in $V(C) \setminus \{u, v\}$ is a tracker. Hence Algorithm 1 gives an optimum tracking set for a chordal graph. □

Lemma 4. ⊛ *Algorithm 1 runs in time $\mathcal{O}(m.n^3)$.*

From Lemma 3 and Lemma 4, we have the following theorem.

Theorem 1. Tracking Paths *can be solved in polynomial time in chordal graphs.*

A similar technique can be used to prove that Tracking Paths is polynomial time solvable for tournament graphs. However, the analysis of tournament graphs is slightly more involved and due to space constraint it is deferred to the full version of the paper [6].

5 Approximation Algorithm and NP-Hardness of Tracking Paths in Bounded-Degree Graphs

In this section, we give an approximation algorithm for Tracking Paths. We show that given an undirected graph G, there exists a polynomial time algorithm that returns a tracking set of the size $2(\delta + 1) \cdot OPT$ for G, where OPT is the size of an optimum tracking set for G and δ is the maximum degree of graph G. Approximation algorithms have been studied for restricted versions of Tracking Shortest Paths and Tracking Paths. Banik et al. gave a 2-approximate algorithm for Tracking Shortest Paths in planar graphs in [2]. Eppstein et al. gave a 4-approximate algorithm for Tracking Paths in planar graphs in [10]. Bilò et al. gave an $\tilde{O}(\sqrt{n})$-approximate algorithm for Tracking Shortest Paths in case of multiple source-destination pairs in [5]. Next we show that Tracking Paths for bounded degree graphs is polynomial time reducible from Vertex Cover for bounded degree graphs.

Lemma 5. *Given an undirected graph G with maximum degree d, there exists an s-t graph G' with maximum degree $2d$, such that G has a vertex cover of size k if and only if G' has a tracking set for all s-t paths, of size $k + |E|^2 + 3|E| - 2$.*

Proof. Let G be and undirected graph with maximum degree d. For reference, let G be the graph in Fig. 2.

We create the graph G' as follows. For each vertex $a \in V(G)$, we introduce a vertex v_a in $V(G')$, and we refer to this set of newly introduced vertices in G' as V_v. For each edge $i \in E(G)$, we introduce two vertices v_i, v_i' in $E(G')$, and we call the set of vertices v_i as V_e, and the set of vertices v_i' as V_e'. The adjacencies between V_v and V_e, V_e' are introduced as follows. If an edge i is incident on vertices a, b in G, then we add edges between the corresponding vertices $v_i, v_i' \in V_e, V_e'$ and the vertices $v_a, v_b \in V_v$ in G'. Next, we add the source and destination vertices s and t in G'. We then create a triangular grid Tg_1 between s and the vertices in V_e, and another triangular grid between the vertices in V_e' and t. See Fig. 3. The vertices of V_v are marked with blank boxes, while the ones from $V_e \cup V_e'$ are marked with solid boxes. The circled vertices form a tracking set. Observe that the maximum degree of vertices in Tg_1 and

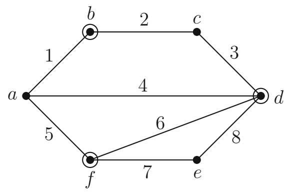

Fig. 2. Depiction of an undirected graph G with maximum degree d

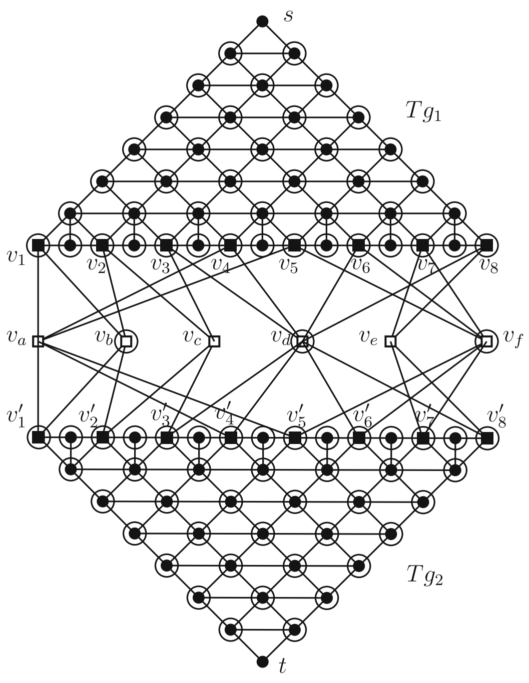

Fig. 3. Depiction of graph G' mentioned in Lemma 5

Tg_2, including the vertices in $V_e \cup V'_e$, is 6. The maximum degree of vertices in V_v is at most $2d$.

Now we prove that there exists a vertex cover of size k in G if and only if there exists a tracking set in G' of size $k + |E|^2 + 3|E| - 2$. First consider the case when G has a vertex cover V_c of size k. We now prove that there exists a tracking set of size $k + |E|^2 + 3|E| - 2$ in G'. We mark the vertices in G' corresponding to V_c as trackers. In addition we mark all the vertices in Tg_1 and Tg_2 (except s and t) as trackers. Now the size of tracking set T in G' is $k + |E|^2 + 3|E| - 2$. We claim that T is a valid tracking set for G'. Suppose not. Then there exists two distinct s-t paths, say P_1, P_2 in G', such that the sequence of trackers in P_1 is same as that in P_2. Observe that two distinct subpaths (subpaths of some s-t paths) contained in Tg_1 (Tg_2) cannot have the same sequence of trackers from $Tg_1 - \{s\}$ ($Tg_2 - \{t\}$). Since all vertices in Tg_1, Tg_2 are marked as trackers, this implies that P_1, P_2 contain the same sequence of vertices from Tg_1 and Tg_2, and they necessarily differ in vertices from V_v. Let $x, y \in V_v$ be the vertices that distinguish P_1 and P_2, and $x \in V(P_1)$ and $y \in V(P_2)$. Since P_1 and P_2 can not differ in their vertex set from Tg_1 and Tg_2, the vertex preceding x, y has to be common in both P_1, P_2. Without loss of generality, we assume that the z is the vertex preceding x, y, and $z \in V(Tg_1)$. This implies that $z \in V_e$. Note that z

corresponds to an edge in G. Since we marked the vertices corresponding to V_c as trackers in G', at least one of the neighbors of z in V_v is necessarily a tracker. Thus either x or y is necessarily a tracker. This contradicts the assumption that P_1 and P_2 have the same sequence of trackers.

Now we consider the case when G' has a tracking set T of size $k+|E|^2+3|E|-2$. We claim that there exists a vertex cover of size k in G. Suppose not. Consider the triangular grid subgraphs Tg_1 and Tg_2. Observe that for each edge (a,b) in Tg_1, there exists a vertex $c \in N(a) \cap N(b)$, and there exists an s-t path, say P_1, that passed through (a,b) in $G-c$, such that we can replace edge (a,b) in P_1 by edges (a,c),(c,b) to form another s-t path, say P_2. Observe that P_1 and P_2 differ in only one vertex i.e. c. Hence c is necessarily a tracker. The same holds true for each edge in Tg_2. Thus all vertices in $V(Tg_1) \cup V(Tg_2) \setminus \{s,t\}$ are necessarily trackers and hence belong to T. Since $|V(Tg_1) \cup V(Tg_2) \setminus \{s,t\}| = |E|^2+3|E|-2$, the remaining k trackers in T are vertices from V_v. Let V_t be the set of vertices in V_v that have been marked as trackers, i.e. $V_t = V_v \cap T$. Note that $|V_t| = k$. We denote the set of vertices in G that correspond to vertices in V_t as V_c. We claim that V_c forms a vertex cover for G. Suppose not. Then there exists an edge, say (a,b) in G, such none of its end points a,b belong to V_c. This implies that the vertices in V_v that correspond to a and b, say v_a, v_B, are not trackers in G'. Due to the construction of G', there exists a pair of vertices $v_i \in V_e$ and $v_i' \in V_e'$ (v_i, v_i' correspond to the edge (a,b) in G) such that v_a and v_b are adjacent to both v_i and v_i'.

Observe that for each pair of vertices v_i, v_i', where $v_i \in V_e$ and $v_i' \in V_e'$, there exists two vertices in V_v (the vertices in $V(G)$ that correspond to the endpoints of the edge i in G) that are adjacent to both v_i and v_i'. Thus for each pair of vertices v_i, v_i', there exists two paths between them passing through two distinct vertices in V_v. Further, there exists a path from s to v_i that is completely contained in Tg_1, and there exists a path from v_i' to t that is completely contained in Tg_2. Thus at least one of the vertices from V_v that are adjacent to v_i, v_i', must necessarily be a tracker. This contradicts the fact that neither v_a nor v_b is a tracker in G'. This completes the proof. □

Since Vertex Cover is known to be NP-hard for graphs with maximum degree d ($d \geq 3$) [12], due to Lemma 5 we have the following corollary.

Corollary 1. Tracking Paths *is* NP*-hard for graphs with maximum degree* $\delta \geq 6$.

Algorithm 2 gives a procedure to find a $2(\delta+1)$-approximate tracking set for undirected graphs with maximum degree δ. We prove its correctness in the following lemma.

Lemma 6. ⊛ *Algorithm 2 gives a* $2(\delta+1)$*-approximate tracking set for an undirected graph.*

Lemma 7. ⊛ *Algorithm 2 runs in time* $\mathcal{O}(n^2)$.

From Lemma 6 and Lemma 7, we have the following theorem.

Algorithm 2: Finding a $2(\delta + 1)$-approximate tracking set for undirected graphs with maximum degree δ.

```
Input: Undirected graph $G = (V, E)$ such that $deg(x) \leq \delta$, $\forall x \in V$, a pair of
       vertices $s, t \in V$.
Output: Tracking Set $T \subseteq V$ for $G$.
1 Apply Reduction Rule 1;
2 Find a 2-approximate feedback vertex set $S$ for $G$;
3 Set $T = S$;
4 foreach $v \in S$ do
5     foreach $x \in N(v)$ do
6         $T = T \cup \{x\}$;
7     end
8 end
9 Return $T$;
```

Theorem 2. *For an undirected graph G on n vertices such that the maximum degree of vertices in G is δ, there exists an $\mathcal{O}(n^2)$ algorithm that finds a $2(\delta+1)$-approximate tracking set for G.*

The approximation ratio for our algorithm can be improved slightly by using the improved approximation bounds known for FVS in bounded degree graphs [1].

6 Reconstructing Paths Using Trackers

In real-world applications, it might be required to identify the s-t path which corresponds to a given sequence of trackers. Banik et al. [2] gave a polynomial time algorithm to reconstruct the shortest s-t path corresponding to a subset of trackers, given a tracking set for shortest s-t paths. Here we give an algorithm which, given a graph G, a constant size tracking set T, and a sequence of trackers π, returns the unique s-t path in G that corresponds to π, if one exists. Our algorithm works for both undirected graphs as well as tournaments.

Theorem 3. *Given a graph G, a tracking set T of constant size k for G, and a sequence of trackers π, the unique s-t path in G corresponding to π, if exists, can be found in polynomial time.*

Proof. Let $V(\pi)$ denote the vertices in the sequence π. Without loss of generality, let $|V(\pi)| = k$ and $\pi = (v_1, v_2, \ldots, v_k)$. Let P be the unique s-t path in G that corresponds to π. Let S be the set of pairs of vertices formed by consecutive vertices in π, preceding and ending with s and t respectively, i.e. $S = \{\{s, v_1\}, \{v_1, v_2\}, \ldots, \{v_k, t\}\}$. Since π is the sequence of trackers in P, $V(P)$ does not contain any trackers from T, other than those in π. In order to find P, we need to find the vertex disjoint paths between each pair of vertices (v_i, v_{i+1}) in S, where $v_0 = s$ and $v_{k+1} = t$. We create a copy v'_i for each vertex v_i in π, and introduce and edge between v'_i and each vertex in $N(v_i)$ in the graph G. Let $S' = \{\{s, v_1\}, \{v'_1, v_2\}, \{v'_2, v_3\} \ldots, \{v_{k-1}, v_k\}, \{v'_k, t\}\}$ and $V(S')$ be the set of all vertices in S'. Consider the graph $G' = G - (T \setminus V(S'))$. If G is an undirected graph, then using the algorithm for disjoint paths in undirected graphs from [15], find the vertex disjoint paths between the pairs of vertices in S', in the graph G'. If G is a tournament graph, then using the algorithm for disjoint paths in tournaments from [8], find the vertex disjoint paths between the pairs of vertices in S', in the graph G'. Since disjoint path problem can be solved in polynomial time for undirected graphs and tournaments [8,15], we can perform this step in polynomial time. Observe that the sequence of these vertex disjoint paths will form an s-t path in G', which will also be an s-t path in G. Note that if the paths between pairs of vertices in S' are not vertex disjoint, it is a violation of the tracking set condition, as there are two vertex disjoint paths between a pair of vertices that have disjoint paths to s and t themselves. Next we prove that the path found will be a unique s-t path. If not, then there exists two s-t paths in G, containing the sequence of trackers π. This contradicts the assumption that T is a tracking set for G. Since T is assumed to be a tracking set for G, if vertex disjoint paths are not found between all pair of vertices in the S', then P does not exist. In this case the algorithm returns NO. $\square$

7 Tracking Edge Set for Undirected Graphs

In this section we study the problem of identifying s-t paths in an undirected edge-weighted graph using the edges of the graph. For a graph G, we define a *tracking edge set* as the set of edges whose intersection with each s-t path results in a unique sequence of edges. Here we allow parallel edges in the input graph. We formally define the problem of tracking paths using edges as follows.

Tracking Paths using Edges (G, s, t)
Input: An undirected edge-weighted graph $G = (V, E)$ with terminal vertices s and t.
Question: Find a minimum weight tracking edge set $T \subseteq E$ for G.

We start by first applying Reduction Rule 1, which ensures that each vertex and edge in the graph participates in some s-t path. Next we prove that each cycle in the reduced graph needs an edge as a tracker.

Lemma 8. ⊛ *For a reduced graph $G = (V, E)$, if $T \subseteq E$ is a tracking edge set, then each cycle in G contains an edge e such that $e \in T$.*

Note that the *tracking set condition* mentioned in Sect. 3 holds for a tracking edge set as well if we consider trackers as edges instead of vertices. Further, by using the arguments similar to those in Lemma 2, the following lemma for tracking using edges (instead of vertices) can be derived. Details are skipped to avoid repetition.

Lemma 9. *In a graph G, if $T \subseteq E(G)$ is not a tracking set for G, then there exist two s-t paths with the same sequence of trackers, and they form a cycle C in G, such that C has a local source a and a local destination b, and $T \cap (E(C) \setminus \{a, b\}) = \emptyset$.*

Next we prove that a feedback edge set (FES) is a tracking edge set for a reduced graph. An FES is a set of edges whose removal makes the graph acyclic.

Lemma 10. *For a reduced graph G, a feedback edge set F is also a tracking edge set for G.*

Proof. Consider graph $G = (V, E)$ reduced by Reduction Rule 1, and an FES $F \subseteq E$ for G. We claim that $T = F$ is a tracking edge set for G. Suppose not. Then there exists two s-t paths, say P_1 and P_2, in G, such that the sequence of tracking edges in both these paths is the same. Due to Lemma 9, the graph induced by P_1 and P_2 contains at least one cycle, say C, such that $C \cap T = \emptyset$. However, since T is an FES for G, it must necessarily contain an edge, say e, from the cycle C marked as a tracking edge. Observe that e can belong to either P_1 and P_2, but not both of them. This contradicts the assumption that P_1 and P_2 contain the same sequence of tracking edges. □

Although finding a minimum FVS is an NP-hard problem, an FES can be found in polynomial time. We now prove that Tracking Paths using Edges can be solved in polynomial time.

Theorem 4. *For an undirected edge-weighted graph G on n vertices,* Tracking Paths using Edges *can be solved in $\mathcal{O}(n^2)$ time.*

Proof. Let G be an undirected edge-weighted graph on n vertices. From Lemma 10 it is known that an FES is a tracking edge set for G. In order to find a minimum weighted tracking edge set for G, we first find a maximum weight spanning tree T for G using Prim's algorithm or Kruskal's algorithm in $\mathcal{O}(n^2)$ time. Now the edges in $G - T$ comprise a minimum weight FES, which is also a minimum weight tracking edge set for G. □

A path reconstruction algorithm similar to the one mentioned in Sect. 6 can be obtained by considering a sequence of tracking edges, and finding vertex disjoint paths between their endpoints in the graph obtained after removal of remaining tracking edges from the tracking edge set for that graph.

8 Conclusion

In this paper, we give polynomial time results for some variants of the Tracking Paths problem. Specifically, we solve Tracking Paths for chordal graphs and tournaments, along with giving an approximation algorithm for degree bounded graphs. We also analyze the problem Tracking Paths using Edges, and prove it to be polynomial time solvable. A constructive algorithm has also been given that helps identify an s-t path, given the unique sequence of trackers it contains. Future scope of this work lies in improving the running times of these algorithms and identifying more graph classes where Tracking Paths may be easily solvable. Open problems include finding approximation algorithms for other NP-hard variants of the problem for both undirected and directed graphs.

Acknowledgement. We thank Prof. Venkatesh Raman for the insightful discussions and suggestions.

References

1. Bafna, V., Berman, P., Fujito, T.: Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In: Staples, J., Eades, P., Katoh, N., Moffat, A. (eds.) ISAAC 1995. LNCS, vol. 1004, pp. 142–151. Springer, Heidelberg (1995). https://doi.org/10.1007/BFb0015417
2. Banik, A., Katz, M.J., Packer, E., Simakov, M.: Tracking paths. In: Fotakis, D., Pagourtzis, A., Paschos, V.T. (eds.) CIAC 2017. LNCS, vol. 10236, pp. 67–79. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57586-5_7
3. Banik, A., Choudhary, P.: Fixed-parameter tractable algorithms for tracking set problems. In: Panda, B.S., Goswami, P.P. (eds.) CALDAM 2018. LNCS, vol. 10743, pp. 93–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74180-2_8
4. Banik, A., Choudhary, P., Lokshtanov, D., Raman, V., Saurabh, S.: A polynomial sized kernel for tracking paths problem. Algorithmica **82**(1), 41–63 (2020)
5. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Tracking routes in communication networks. In: Censor-Hillel, K., Flammini, M. (eds.) SIROCCO 2019. LNCS, vol. 11639, pp. 81–93. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24922-9_6
6. Choudhary, P.: Polynomial time algorithms for tracking path problems. CoRR abs/2002.07799 (2020). https://arxiv.org/abs/2002.07799
7. Choudhary, P., Raman, V.: Improved kernels for tracking path problems. CoRR abs/2001.03161 (2020). http://arxiv.org/abs/2001.03161
8. Chudnovsky, M., Scott, A., Seymour, P.: Disjoint paths in tournaments. Adv. Math. **270**, 582–597 (2015)
9. Duraisamy, K., Dempsey, K., Ali, H., Bhowmick, S.: A noise reducing sampling approach for uncovering critical properties in large scale biological networks. In: 2011 International Conference on High Performance Computing Simulation, pp. 721–728, July 2011. https://doi.org/10.1109/HPCSim.2011.5999898
10. Eppstein, D., Goodrich, M.T., Liu, J.A., Matias, P.: Tracking paths in planar graphs. In: 30th International Symposium on Algorithms and Computation, ISAAC 2019, 8–11 December 2019, Shanghai University of Finance and Economics, Shanghai, China, pp. 54:1–54:17 (2019)
11. Fisher, D.C., Ryan, J.: Tournament games and condorcet voting. Linear Algebra Appl. **217**, 87–100 (1995). Proceedings of a Conference on Graphs and Matrices in Honor of John Maybee
12. Garey, M., Johnson, D., Stockmeyer, L.: Some simplified np-complete graph problems. Theoret. Comput. Sci. **1**(3), 237–267 (1976)
13. Geman, D.: Random fields and inverse problems in imaging. In: Hennequin, P.-L. (ed.) École d'Été de Probabilités de Saint-Flour XVIII - 1988. LNM, vol. 1427, pp. 115–193. Springer, Heidelberg (1990). https://doi.org/10.1007/BFb0103042

14. Golumbic, M.C.: Perfect graphs (chapter 3). In: Golumbic, M.C. (ed.) Algorithmic Graph Theory and Perfect Graphs, pp. 51–80. Academic Press (1980)
15. Kawarabayashi, K., Kobayashi, Y., Reed, B.: The disjoint paths problem in quadratic time. J. Comb. Theory Ser. B **102**(2), 424–435 (2012)
16. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. J. Roy. Stat. Soc.: Ser. B (Methodol.) **50**(2), 157–224 (1988)
17. McGarvey, D.C.: A theorem on the construction of voting paradoxes. Econometrica **21**(4), 608–610 (1953)
18. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989). https://doi.org/10.1109/5.18626
19. Stearns, R.: The voting problem. Am. Math. Mon. **66**(9), 761–763 (1959)