Algorithmica (2022) 84:1548–1570
https://doi.org/10.1007/s00453-022-00931-1

Polynomial Time Algorithms for Tracking Path Problems

Pratibha Choudhary[1]

Received: 4 August 2020 / Accepted: 7 January 2022 / Published online: 8 February 2022

Abstract
Given a graph G, and terminal vertices s and t, the TRACKING PATHS problem asks to compute a set of 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 undirected chordal graphs and tournament graphs. We also show that TRACKING PATHS is NP-hard in graphs with bounded maximum degree $\Delta \geq 6$, and give a $2(\Delta+1)$-approximate algorithm for this case. Further, we give a polynomial time 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. Finally we 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.

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

A preliminary version of the paper appeared in the proceedings of IWOCA 2020 [10].
P. Choudhary: Part of this work was done while the author was affiliated with Indian Institute of Technology Jodhpur, Jodhpur, India and was visiting The Institute of Mathematical Sciences, Chennai, India.

✉ Pratibha Choudhary
pratibhac247@gmail.com

[1] Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

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, and networking systems.

The problem of tracking paths in a network was first graphically modeled by Banik et al. in [3]. 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 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. [3] 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 [5,11,15]. 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 [5]. Here, the authors studied the parameterized version of TRACKING PATHS, which asks if there exists a tracking set of size at most k, and proved it fixed-parameter tractable (FPT) when parameterized by k, by showing that the problem admits 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 [11], the authors improved this kernel to $\mathcal{O}(k^2)$, and gave an $\mathcal{O}(k)$ kernel for planar graphs. In [15], 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 [4,8]. In [4], Banik et al. studied TRACKING SHORTEST PATHS and proved the problem fixed-parameter tractable. In [8], Bilò et al. proved that TRACKING SHORTEST PATHS is NP-hard for cubic planar graphs in case of multiple source-destination pairs, and gave an FPT algorithm parameterized by the number of vertices equidistant from the source s. A related model of the problem was studied in [7] where the author focuses on differentiating walks in directed graphs using arcs.

In this paper we study TRACKING PATHS for chordal graphs, tournament graphs, and bounded degree 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 unique 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 [14,19,22,24]. Tournament graphs are used in voting theory and social choice theory to graphically depict pairwise relationships between entities in a community [23,27]. Tournament graphs are particularly used to study the Condorcet voting model, where a preference is indicated between each pair of contestants [16]. We would like to point out that all the results in the paper hold for undirected graphs, except for the case when we refer to tracking paths in tournaments.

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, then that vertex necessarily needs to be marked as a tracker. Next we show 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 [5]. 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, a 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 [21] and tournaments [12] to construct the required s-t path if one exists.

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 a 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 to 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 in an undirected graph. In a directed graph, (a, b) represents an edge directed from vertex a towards vertex b. For a graph G, $V(G)$ represents the vertex set of G and $E(G)$ represents the edge set of G. A graph G' is called a subgraph of G, if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G) \cap \{(a, b) \mid a, b \in V(G')\}$. A graph G' is called an induced subgraph

of G if $V(G') \subseteq V(G)$ and $E(G') = E(G) \cap \{(a, b) \mid a, b \in V(G')\}$. For a graph $G = (V, E)$ and a set of vertices $V' \subseteq V$, we use $G(V')$ to denote the subgraph induced by V', i.e. $G' = (V', \{(a, b) \mid a, b \in V'\} \cap E)$. For a graph G, the *neighborhood* of a vertex $v \in V(G)$ is denoted by $N(v) = \{x \mid (x, v) \in E(G)\}$. In case of directed graphs, $N^+(v) = \{x \mid (x, v) \in E\}$ is referred to as the *out-neighbourhood* of vertex v, and $N^-(v) = \{x \mid (v, x) \in E\}$ is referred to as the *in-neighbourhood* of vertex v. We use $deg(v) = |N(v)|$ to denote the degree of vertex v. 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 $S = \{x\}$. If $(a, b) \in E$, then $G - (a, b)$ denotes the graph formed by the vertex set $V(G)$ and the edge set $E - (a, b)$. A *chord* in a cycle is an edge between two vertices of the cycle, such that the edge itself is not part of the cycle. In a directed graph, a *monotone* cycle is a subgraph C which comprises two distinct directed paths between a pair of vertices $a, b \in V$. Except for the case of tournaments, by *graph* we mean an undirected graph.

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 endpoints, i.e. $V(P_1) \cap V(P_2) \subseteq \{a, b\}$, where a and b are the starting and endpoints of the paths. By distance we mean the length of a 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). In a directed graph, by a path we mean a directed path. 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 Δ, where Δ 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 [5] that ensures that each vertex and edge in the input graph participates in an s-t path.

Reduction Rule 1 [5] *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 [5]. 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 [5], 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 recall the *tracking set condition*, which is useful for validation of a tracking set [5]. Specifically, it is known from [5], 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*:

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 $G((V \setminus (T \cup \{s, t\})) \cup \{u, v\})$, and
- there exists a path from s to u, say P_{su}, and a path from v to t, say P_{vt}, in $G((V \setminus (V(P_1) \cup V(P_2)) \cup \{u, v\})$, 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.

Next, we use the tracking set condition to prove the following lemma. Although the main idea of the lemma has been discussed with a different perspective in [5], we give the proof here for completeness.

Lemma 2 *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 $T \cap (V(C) \setminus \{a, b\}) = \emptyset$.*

Proof Let G be a graph, such that $T \subseteq V(G)$ is not a tracking set for G. Due to the tracking set condition, it is known that in such a case, there exist two distinct vertices u, v along with two distinct paths P_1, P_2 between u and v, such that there are no trackers on P_1 and P_2 except possibly at u and v. Further, there exists a path P_{su} from s to u and a path P_{vt} from v to t, and such that these P_{su} and P_{vt} are vertex disjoint, and they intersect with P_1 and P_2 only at u and v. Let G' be the graph induced by $V(P_1) \cup V(P_2)$. Observe that u and v form a local source-destination pair for G'. Note that no vertex, other than possibly u and v, in G' is a tracker. Starting from u, let a be the last vertex in G' until which paths P_1 and P_2 have the same sequence of vertices. Let $b \in V(P_1)$ be the first vertex in P_1 after a, such that $b \in V(P_1) \cap V(P_2)$. We use P_{ab_1} to denote the subpath of P_1 lying between vertices a and b, and P_{ab_2} to denote the subpath of P_2 lying between the vertices a and b. Observe that paths P_{ab_1} and P_{ab_2} are vertex disjoint (except for vertices a and b) and thus form a cycle, say C. Further there exists a subpath of P_2 between b and v, that intersects C only at b. Since P_1 and P_2 share the same vertex sequence from u to a, a is a local source for C. Also, by construction, b is a local destination for C. Note that it is possible that $u = a$ and/or

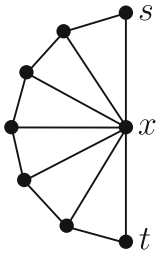

Fig. 1 Depiction of a chordal graph on n vertices with an optimum tracking set ($V(G)\backslash\{s,t\}$) of size $n-2$ and an FVS (vertex x) of size 1

$b = v$. However, $V(C)\backslash\{a,b\}$ does not contain any trackers. Hence the lemma holds. □

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.

4.1 Tracking Paths in 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 [20].

In undirected graphs, a tracking set is also a feedback vertex set [5]. However, a tracking set can be arbitrarily larger in size compared to a feedback vertex set. This holds true for chordal graphs as well. See Fig. 1. It can be seen that here an FVS can consist of a single vertex x, whereas a minimum tracking set consists of all vertices in $V\backslash\{s,t\}$, where V is the vertex set of the graph.

Algorithm 1: Finding a 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.

```
Initialize T = ∅; Apply Reduction Rule 1;
foreach e = (a, b) ∈ E do
    foreach x ∈ (N(a) ∩ N(b))\T do
        if ∃ an s-t path P in G − x such that e ∈ E(P) then
            T = T ∪ {x};
        end
    end
end
Return T;
```

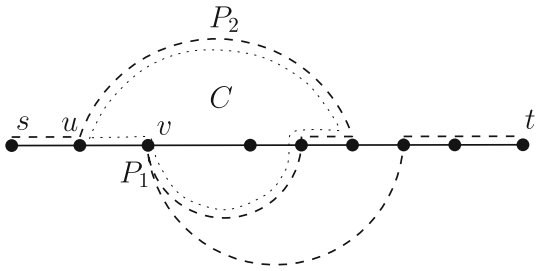

Fig. 2 Indistinguishable s-t paths in a graph form a cycle (marked in dotted lines)

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.*

Proof Algorithm 1 first ensures that each vertex and edge in the input graph $G = (V, E)$ participates in an s-t path. Next for each edge $e = (a, b) \in E$, if there exists a vertex $x \in (N(a) \cap N(b)) \backslash T$, the algorithm checks if there exists an s-t path in $G - x$ that contains the edge e. Let P be 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 needs to belong to a tracking set for G. This proves the optimality of the algorithm, since the vertices we mark as trackers, necessarily need to be trackers.

Now we prove that Algorithm 1 indeed returns a tracking set T for G. Suppose not. Then T is not a tracking set for G. Due to Lemma 2, there exist two s-t paths, say P_1, P_2, that form a cycle C in G, such that C has a local source u and a local destination v, and $V(C) \backslash \{u, v\}$ does not contain any trackers. See Fig. 2. 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 $V(C) \backslash \{u, v\}$ can be a tracker. Since we consider graphs without any parallel edges, there exists at least one vertex in $V(C) \backslash \{u, v\}$.

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

Next, consider the case when C is not a triangle, i.e. C contains four or more vertices. Since G is a chordal graph, C contains a chord. Now we show that it is possible to find a triangle R containing a chord in C, such that at least one of the vertices from $V(R) \cap (V(C) \backslash \{u, v\})$ has been marked as a tracker, thus distinguishing P_1 and P_2.

Suppose w and x are two vertices such that $(u, w), (u, x) \in E(C)$. Now we consider the two cases when $(w, x) \in E$ and $(w, x) \notin E$.

1. $(w, x) \in E$.
   Without loss of generality, let $x \in V(P_1)$ and $w \in V(P_2)$. Observe that the edge (u, x) in path P_1 can be replaced by the path formed by concatenation of edges $(u, w) \cdot (w, x)$, 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. Further, observe that the edge (u, w) in path P_2 can be replaced by the concatenated path $(u, x) \cdot (x, w)$, to obtain a new path that differs from P_2 only at the vertex x. Hence x must have been

marked as a tracker by Algorithm 1. Let $(x, y) \in E(C)$, i.e. y is the neighbour of x in the cycle C.

2. $(w, x) \notin E$.
   Observe that since G is a chordal graph, if $(w, x) \notin E$, then necessarily $(u, y) \in E$. Now observe that the edge (u, y) in path P_1 can be replaced by the path formed by the concatenation of edges $(u, x) \cdot (x, y)$ to form a new path, such that these two paths differ only in vertex x. Thus x must have been marked as a tracker by Algorithm 1.

The above cases contradict the assumption that no vertex in $V(C)\backslash\{u, v\}$ is a tracker. Hence Algorithm 1 gives an optimum tracking set for a chordal graph. □

Next we prove that Algorithm 1 runs in polynomial time.

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

Proof Let $G = (V, E)$ be the input graph, $|V| = n$ and $|E| = m$. Due to [5], it is known that it takes $\mathcal{O}(n^2)$ (quadratic) time to apply Reduction Rule 1. Next for each edge $e = (a, b) \in E$, we consider the set of vertices that are adjacent to both endpoints a, b of the edge e. This takes $\mathcal{O}(m \cdot n)$ time. Now for each vertex x that is adjacent to both a and b, we check if e participates in some s-t path in the graph $G - x$. Removal of vertex x from G takes $\mathcal{O}(n)$ time. In order to check if e participates in some s-t path in $G - x$, we check if there exists a path between s and a, say P_1, and a path between b and t, say P_2, such that $V(P_1) \cap V(P_2) = \emptyset$. This can be done using the algorithm for finding vertex disjoint paths shown in [21] in $\mathcal{O}(n^2)$ time. Thus the total time taken to run Algorithm 1 is $\mathcal{O}(n^2) + \mathcal{O}(m \cdot n^2 \cdot n^2)$, i.e. $\mathcal{O}(m \cdot n^4)$. □

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

Theorem 1 *In a chordal graph with n vertices and m edges,* Tracking Paths *can be solved in $\mathcal{O}(m \cdot n^4)$ time.*

4.2 Tracking Paths in Tournaments

Recall that tournaments are directed graphs that have a directed edge between each pair of vertices in the graph. A lot of problems including Feedback Vertex Set and Feedback Arc Set are known to be NP-hard in tournaments [9,26]. It is known that Tracking Paths is NP-hard for directed acyclic graphs [6]. This implies that Tracking Paths is NP-hard for directed graphs as well. However, as we prove now, Tracking Paths is in P for tournament graphs. We start by first applying a slight variation of Reduction Rule 1, to ensure that each vertex and edge in the input graph participates in an s-t path. Note that although deletion of a vertex does not affect the graph properties, deletion of an edge might result in a graph that is not a tournament graph anymore. Hence, we give a slightly modified reduction rule for the case of tournaments.

Reduction Rule 2 *In a graph G, if there exists a vertex that does not participate in any s-t path in G, then delete it. If there exists an edge that does not participate in any s-t path in G, then mark it useless.*

Now we prove that Reduction Rule 2 is safe and can be applied on tournament graphs in polynomial time.

Lemma 5 *Reduction Rule 2 is safe and can be applied in polynomial time in tournament graphs.*

Proof Consider a tournament graph $G = (V, E)$. If a vertex or an edge in G does not participate in any s-t path, then it can not contribute to tracking any paths, nor does it need to be considered while ensuring that each s-t path has a unique sequence of trackers. Hence, Reduction Rule 2 is safe.

In order to apply Reduction Rule 2, for each edge $e = (a, b) \in E$, we check if there exists a path from s to a, say P_1, and a path from b to t, say P_2, such that $V(P_1) \cap V(P_2) = \emptyset$, using the algorithm for finding vertex disjoint paths in tournaments given in [12] in $\mathcal{O}(n^2)$ time. If such paths do not exist, then mark edge e as useless. After repeating the process for all edges in G, we delete the vertices for which all incident edges have been marked useless. Thus Reduction Rule 2 can be applied in $\mathcal{O}(m \cdot n^2)$ time. □

Algorithm 2: Finding a Tracking Set for a Tournament Graph.

Input: Tournament graph $G = (V, E)$ and vertices $s, t \in V$.
Output: Tracking Set $T \subseteq V$ for G.

```
Initialize T = ∅;
Apply Reduction Rule 2;
foreach e = (a, b) ∈ E do
    foreach x ∈ (N⁺(a) ∩ N⁻(b))\T do
        if ∃ an s-t path P in G − x and e ∈ E(P) then
            T = T ∪ {x};
        end
    end
end
Initialize G′ = G − T;
foreach (u, v) ∈ E(G) do
    if ∃ a directed path P from u to v in G′ − (u, v) then
        if ∃ an s-t path P′ in G − (V(P) − {u, v}) and (u, v) ∈ E(P′) then
            Let x be an arbitrary vertex in V(P) − {u, v};
            T = T ∪ {x};
            G′ = G − T;
        end
    end
end
Return T;
```

Algorithm 2 gives a procedure to compute a minimum tracking set for a tournament graph G. In the upcoming lemmas, we prove the correctness, optimality and the running time of the algorithm. In order to do so, we define a few terms. We use *t-monotone cycle* to denote a monotone cycle that needs a tracker, i.e. a monotone cycle that forms two distinct s-t paths. C is a t-monotone cycle in Fig. 4. The vertex with two outgoing

edges in a t-monotone cycle is called its *entry vertex* and the vertex with two incoming edges in a t-monotone cycle is called its *exit vertex*.

Lemma 6 *Algorithm 2 returns a tracking set for a tournament graph.*

Proof We start with a brief summary of the algorithm. Let $G = (V, E)$ be an input graph. In the first part of the algorithm, for each edge $e = (a, b) \in E$, if there exists a vertex $x \in (N^+(a) \cap N^-(b)) \backslash T$, we check if there exists an s-t path in $G - x$ that contains the edge e. If yes, then we mark x as a tracker. In the second part of the algorithm, for each edge $(u, v) \in E$, we check if there exists a directed path P from u to v without any trackers. If yes, we check if there exists a s-t path containing (u, v) in the graph induced by $V \backslash V(P)$. If such a s-t path is found, then we arbitrarily mark a vertex in $V(P) \backslash \{u, v\}$ as a tracker.

We claim that Algorithm 2 indeed returns a tracking T set for G. Suppose not. Then there exist two s-t paths, say P_1 and P_2, that contain the same sequence of trackers in G. Consider the graph G' induced by $V(P_1) \cup V(P_2)$. Starting from s, let u be the last vertex until which P_1 and P_2 contain the same sequence of vertices. Let $v \in V(P_1)$ be the first vertex on P_1 after u, such that $v \in V(P_2)$. Let $P_{1[u,v]}$ be the subpath of P_1 lying between the vertices u and v, and $P_{2[u,v]}$ be the subpath of P_2 lying between the vertices u and v. Observe that $P_{1[u,v]}$ and $P_{2[u,v]}$ are vertex disjoint except for their endpoints u and v, hence they form a monotone cycle, say C. Further, there exists a subpath of P_2 from v to t that intersects with C only at v. Thus C is a t-monotone cycle. See Fig. 3, where path P_1 is marked in dashed lines, while path P_2 is marked in solid lines. Observe that if T is not a tracking set then C is a t-monotone cycle, i.e. it is a monotone cycle that needs a tracker. In the remaining part of the proof, we show that such a cycle cannot exist.

We consider the different cases based on the lengths of paths $P_{1[u,v]}$ and $P_{2[u,v]}$. For simplicity of notation, we denote the two distinct paths between the entry and exit vertices in C by P_1' and P_2''. Since there are no parallel edges in G, the length of at least one of these paths is greater than one.

1. The length of P_1' is 1 and the length of P_2'' is 2:
   In this case C is a triangle. However, note that the algorithm ensures that each monotone cycle that is a triangle is marked with a tracker at the vertex that is not the entry or exit vertex of that triangle. This contradicts the assumption that C is a t-monotone cycle.
2. The length of P_1' is 1 and that P_2'' is greater than 2:
   In this case C is a monotone cycle with P_1' being an edge and P_2'' being a directed path from u to v. However, note that in the second part of the algorithm, such a cycle is marked with a tracker on a vertex in $V(C) \backslash \{u, v\}$. This is a contradiction to the assumption that C is a t-monotone cycle.
3. Length of both P_1' and P_2'' is greater than 2:
   Let x be the first vertex in P_1' after u and y be the first vertex in P_2'' after u. Since G is a tournament either $(x, y) \in E$ or $(y, x) \in E$. See Fig. 4. First we consider the case when $(x, y) \in E$. Observe that paths P_2 and $P_2' \cdot (u, x) \cdot (x, y) \cdot P_2'' - (u, y) \cdot P_2'''$ are two s-t paths that lead to formation of a t-monotone triangle such that the paths differ only in vertex x. Hence x must have been marked as a tracker by

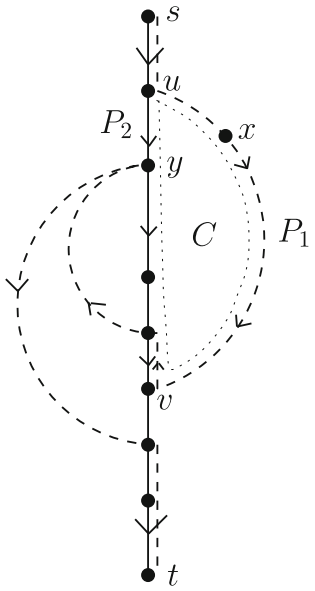

Fig. 3 Indistinguishable s-t paths in a graph form a cycle (marked in dotted lines)

Algorithm 2. The case when $(y, x) \in E$ is symmetric since the paths $P_2' \cdot P_1' \cdot P_2'''$ and $P_2' \cdot (u, y) \cdot (y, x) \cdot P_1' - (u, x) \cdot P_2'''$ also lead to formation of a t-monotone triangle such that the paths differ only in vertex y. This contradicts the assumption that C is a t-monotone cycle.

Hence, it holds that after applying Algorithm 2 there are no t-monotone cycles in G. Thus Algorithm 2 returns a tracking set for G. □

Lemma 7 *Algorithm 2 returns an optimum tracking set for a tournament graph.*

Proof Let $G = (V, E)$ be the input graph. In the first part of the algorithm, for each edge $(a, b) \in E$, we check if there exists a vertex $x \in N^+(a) \cap N^-(b)$. If such a vertex exists, we check if (a, b) participates in an s-t path, say P, in the graph $G - x$. If yes, then x is marked as a tracker. Observe that the edge (a, b) in P can be replaced by the edges $(a, x), (x, b)$ to form a new path, say P_x, such that P and P_x differ in exactly the vertex x. Hence, any tracking set for G necessarily requires x to be marked as a tracker. Thus all vertices marked as trackers in the first part of the algorithm shall be present in every optimal tracking set for the graph.

In the second part of the algorithm, for each edge $(u, v) \in E$, we check if there exists a directed path from u to v in the graph $G - (u, v)$. If such a path, say P, exists then we further check if (u, v) belongs to an s-t path in the graph $G - V(P)$. If yes, then a vertex is chosen arbitrarily from $P \backslash \{u, v\}$ and marked as a tracker. Recall that since we consider graphs without any parallel edges, there exists at least one such vertex in $V(C) \backslash \{u, v\}$. In the rest of the proof, we show that this step is also optimum.

Let P_1 and P_2 be two s-t paths that are indistinguishable after the first part of the algorithm i.e. after all the triangles are appropriately marked with trackers. Let C be a

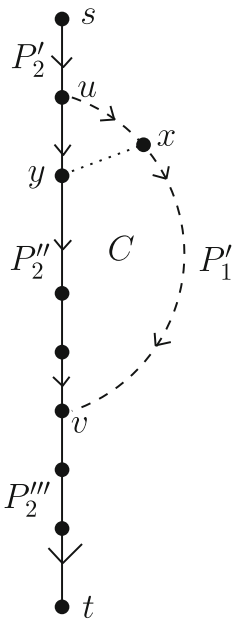

Fig. 4 Monotone cycle

monotone cycle formed due to paths P_1 and P_2 with entry vertex u and exit vertex v. We use P_2' to denote the subpath of P_2 lying between vertices s and u, P_2'' to denote the subpath of P_2 between vertices u and v, and P_2''' to denote the subpath of P_2 between vertices v and t. P_1' denotes the subpath of P_1 between vertices u and v. Thus $V(C) = V(P_1') \cup V(P_2'')$. See Fig. 4.

Now we analyze different cases based on the lengths of paths P_1' and P_2''.

1. *The length of both* P_1' and P_2'' is 1:
   Since we do not consider parallel edges, such a case is not possible.
2. *The length of one among* P_1' *and* P_2'' *is* 1 *and the other is* 2:
   This leads to the formation of a triangle but all t-monotone triangles have already been marked with trackers. Hence, such a case is not possible.
3. *The lengths of both* P_1' *and* P_2'' *are at least* 2:
   Let x, y be the two out-neighbors of vertex u such that $x, y \in V(C)$ and $x \in V(P_1)$ and $y \in V(P_2)$. Since G is a tournament, there exists an edge between the vertices x and y. First we consider the case when $(x, y) \in E$. Observe that paths P_2 and $P_2' \cdot (u, x) \cdot (x, y) \cdot (P_2'' - (u, y)) \cdot P_2'''$ are two s-t paths that lead to formation of a t-monotone triangle such that the paths differ only in vertex x. Hence x must have been marked as a tracker by Algorithm 2. The case when $(y, x) \in E$ is symmetric since the paths $P_2' \cdot P_1' \cdot P_2'''$ and $P_2' \cdot (u, y) \cdot (y, x) \cdot (P_1' - (u, x)) \cdot P_2'''$ also lead to formation of a t-monotone triangle such that the paths differ only in vertex y. This contradicts the assumption that P_1 and P_2 have the same sequence of trackers.
4. *The length of one among* P_1' *and* P_2'' *is* 1, *and the length of the other path is at least* 3: Without loss of generality, let the length of P_2'' be one and the length of

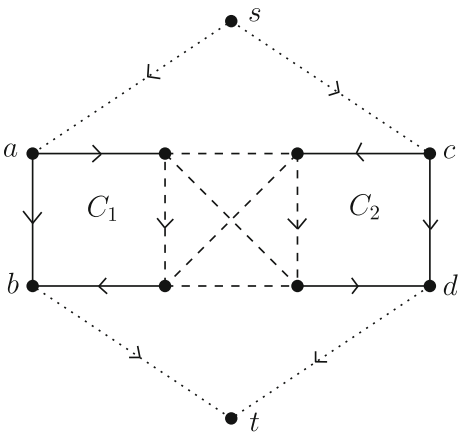

Fig. 5 Two disjoint t-monotone cycles. The dotted lines denote paths to s and t, the solid lines indicate the t-monotone cycles, and the dashed lines indicate the formations of t-monotone triangles

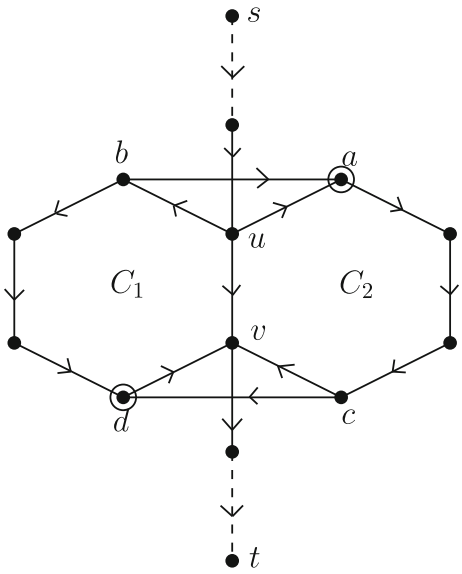

Fig. 6 Two t-monotone cycles sharing the edge side, circled vertices are trackers

P_1' be at least three. We refer to the edge that forms P_2'' as the *edge-side* and P_1' as the *long side* of cycle C. Also for the rest of the proof, by marking a vertex in a cycle as a tracker, we mean marking a vertex other than the entry/exit vertex in the cycle as a tracker. Now we shall prove that once a tracker is marked in each t-monotone cycle that is a triangle, for the remaining t-monotone cycles, we can arbitrarily mark a vertex as a tracker. Note that the only exception to this is when two t-monotone cycles overlap and choosing a tracker arbitrarily is not optimal. Next we consider the possible ways in which two distinct t-monotone cycles, say C_1 and C_2, may overlap.

– C_1 and C_2 are vertex disjoint: In such a case there shall be a t-monotone triangle formed between vertices of the two cycles. See Fig. 5. (a, b) is the edge side of

cycle C_1 and (c, d) is the edge side of cycle C_2. This implies that at least one t-monotone triangle is formed by vertices from the long sides (depicted by dashed lines) of C_1 and C_2. Since all t-monotone triangles have already been marked as trackers, such a case is not possible.

- C_1 and C_2 share only the edge side: A t-monotone triangle is formed in a way that at least one of the cycles will have a tracker on a vertex other than the entry/exit vertices. See Fig. 6. Cycles C_1 and C_2 are two t-monotone cycles that share the edge-side (u, v). Circled vertices represent trackers. Note that if $(b, a), (c, d) \notin E$ and $(a, b), (d, c) \in E$, then b, d would have been marked as trackers. Hence, such a case is not possible.
- C_1 and C_2 share all the vertices on the long side: If the sequence of vertices is different in C_1 and C_2, then there shall be a t-monotone triangle in the cycle such that at least one of the cycles among C_1 and C_2 contains a tracker. Else, if the sequence of vertices is same for C_1 and C_2 in the long side, then we can arbitrarily pick any vertex from $V(C_1) \cap V(C_2)$ as a tracker.
- An edge from the long side of C_1 is the edge side of C_2: In this case at least two trackers are required, one in the long side of each of the cycles. Since the long sides of the cycles do not overlap, we can arbitrarily choose one vertex from the long side of each of C_1 and C_2.
- C_1 and C_2 share some (but not all) vertices from the long side: There exists a pair of vertices $x, y \in V(C_1) \cap V(C_2)$ such that there is a directed path from x to y which is also a subgraph of C_1 and C_2. Since the cycles do not share all vertices on the long side, there exists a pair of vertices a, b adjacent to either x or y, such that $a \in V(C_1) \backslash V(C_2)$ and $b \in V(C_2) \backslash V(C_1)$. Without loss of generality, let $a, b \in N^-(x)$. Since G is a tournament graph, there exists an edge between a, b, leading to the formation of a t-monotone triangle. Thus at least one among a or b must have already been marked as a tracker by the algorithm. Hence, such a case is not possible.

In the above we have shown that after all the t-triangles are marked with trackers in the first part of the algorithm, the only vertex disjoint paths that can be left untracked are the ones in which at least one path in the corresponding t-monotone cycle is an edge. Further, we show that a tracker can be marked arbitrarily in the long side of such a cycle. Hence, the algorithm returns an optimum tracking set. □

Lemma 8 *Algorithm 2 runs in $\mathcal{O}(n^6)$ time.*

Proof Let $G = (V, E)$ be the input graph, $|V| = n$ and $|E| = m$. For each edge $e = (a, b) \in E$, we consider the set of vertices that are adjacent to both endpoints a, b of the edge e. This takes $\mathcal{O}(m \cdot n)$ time. Now for each vertex x that is adjacent to both a and b, we check if e participates in some s-t path in the graph $G - x$. Removal of vertex x from G takes $\mathcal{O}(n)$ time. In order to check if e participates in some s-t path in $G - x$, we check if there exists a path between s and a, say P_1, and a path between b and t, say P_2, such that $V(P_1) \cap V(P_2) = \emptyset$. This can be done using the algorithm for finding vertex disjoint paths in tournaments shown in [12] in $\mathcal{O}(n^2)$ time. The steps in the second part of the algorithm can also be carried out using the disjoint paths algorithm [12]. Thus Algorithm 2 runs in $\mathcal{O}(n^6)$ time. □

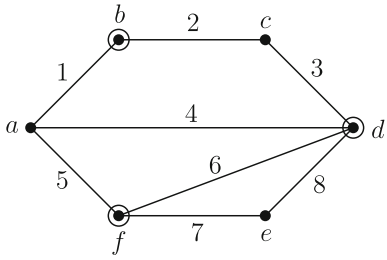

Fig. 7 Depiction of an undirected graph G with maximum degree $\Delta \geq 3$

From Lemmas 6 and 8, we have the following theorem.

Theorem 2 TRACKING PATHS *can be solved in polynomial time in tournament graphs.*

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 [3]. Eppstein et al. gave a 4-approximate algorithm for TRACKING PATHS in planar graphs in [15]. Bilò et al. gave an $\tilde{\mathcal{O}}(\sqrt{n})$-approximate algorithm for TRACKING SHORTEST PATHS in case of multiple source-destination pairs in [8]. Next we show that TRACKING PATHS for bounded degree graphs is polynomial time reducible from VERTEX COVER for bounded degree graphs.

Lemma 9 *Given an undirected graph G with maximum degree Δ, there exists an s-t graph G' with maximum degree 2Δ, 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 an undirected graph with maximum degree Δ. For reference, let G be the graph in Fig. 7. The graph G has vertices labeled from a to f and edges numbered from 1 to 8.

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 with 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. 8. 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

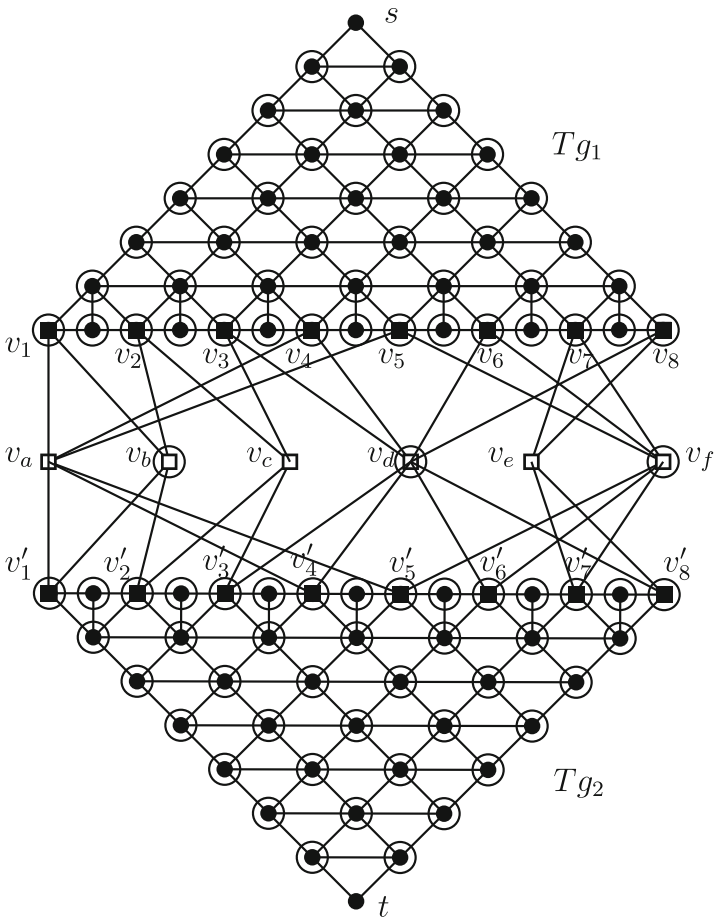

Fig. 8 Depiction of graph G' mentioned in Lemma 9

vertices in Tg_1 and Tg_2, including the vertices in $V_e \cup V'_e$, is 6. The maximum degree of vertices in V_v is at most 2Δ.

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 exist two distinct s-t paths, say P_1, P_2 in G', such that the sequence of trackers in P_1 is the 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 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 $V(Tg_1) \cup V(Tg_2)\backslash\{s, t\}$ are necessarily trackers and hence belong to T. Since $|V(Tg_1) \cup V(Tg_2)\backslash\{s, t\}| = |E|^2 + 3|E| - 2$, 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 endpoints 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 exist 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 exist 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 Δ ($\Delta \geq 3$) [18], due to Lemma 9 we have the following corollary.

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

Algorithm 3 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 10 *Algorithm 3 gives a* $2(\Delta + 1)$*-approximate tracking set for an undirected graph.*

Proof Algorithm 3 starts by ensuring that each vertex and edge in the input graph G participates in an s-t path of G by applying Reduction Rule 1.

Algorithm 3: Finding a $2(\Delta + 1)$-approximate tracking set for undirected graphs with maximum degree Δ.

Input: Undirected graph $G = (V, E)$ such that $\forall x \in V, deg(x) \leq \Delta$, a pair of vertices $s, t \in V$.
Output: Tracking Set $T \subseteq V$ for G.

```
Apply Reduction Rule 1;
Find a 2-approximate feedback vertex set S for G;
Set T = S;
foreach v ∈ S do
    foreach x ∈ N(v) do
        T = T ∪ {x};
    end
end
Return T;
```

Next we claim that Algorithm 3 indeed returns a tracking set T for G. Suppose not. Then due to Lemma 2, there exists a cycle C in G with a local source u and a local destination v, such that $V(C)\backslash\{u, v\}$ does not contain any trackers. See Fig. 2, where path P_1 is marked in solid lines, while path P_2 is marked in dashed lines.

Observe the cycle C formed due to paths P_1 and P_2. Since P_1 and P_2 contain the same sequence of trackers, no vertex in $V(C)\backslash\{u, v\}$ can be a tracker. Since we consider graphs without any parallel edges, there exists at least one vertex in $V(C)\backslash\{u, v\}$. Note that Algorithm 3 includes a 2-approximate feedback vertex set, S, for G in T. Thus at least one vertex from C belongs to T. Note that it is possible that vertices $u \in S$ and/or $v \in S$, and thus u or v may have been included in T. But the vertices u, v do not help distinguish between paths P_1 and P_2. However, observe that Algorithm 3 also includes all neighbors of the vertices in S into T. Further each vertex in $V(C)$ has at least two of its neighbors in $V(C)$. Thus at least one vertex in $V(C)$, other than u and v, will have been necessarily included in T. This violates the claim that no vertex other than u or v belongs to T, contradicting the assumption that T is not a tracking set for G.

Next we explain the approximation ratio $2(\Delta + 1)$. Let G be an input graph and x be the size of a minimum feedback set for G. Let T be the tracking set computed by Algorithm 3. When the algorithm includes a 2-approximate feedback vertex set S into T, the size of T is at most $2 \cdot x$. Further, for each vertex in S, all of its neighbors are also included into the tracking set T. Since the maximum degree of G is upper bounded by Δ, for each vertex in S, additional Δ vertices are included in T. Thus the final size of T is at most $2(\Delta + 1) \cdot x$. From [5], we know that the size of a tracking set is at least the size of a minimum tracking set. Hence, the size of T is at most $2(\Delta + 1) \cdot OPT$, where OPT is the size of an optimum tracking set for G. □

Next we prove that Algorithm 3 runs in polynomial time.

Lemma 11 *Algorithm 3 runs in time $\mathcal{O}(n^2 \log n)$.*

Proof Algorithm 3 starts by applying Reduction Rule 1 that can be applied in quadratic time. Next we find a 2-approximate feedback vertex set S for the input graph, using the algorithm given in [1] in $\mathcal{O}(\min\{|E| \log |V|, |V|^2\})$ time. We include S in the tracking

set T. Next, for each vertex $v \in S$, we add $N(v)$ to T. This step takes $\mathcal{O}(n^2 \log n)$ time. Thus the overall time taken is $\mathcal{O}(n^2) + \mathcal{O}(\min\{|E| \log |V|, |V|^2\}) + \mathcal{O}(n^2 \log n)$. Hence the algorithm runs in total $\mathcal{O}(n^2 \log n)$ time. □

From Lemmas 10 and 11, we have the following theorem.

Theorem 3 *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 bound known for FVS in bounded degree graphs [2] which has a performance ratio of $2 - \frac{2}{3\Delta - 2}$.

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. [3] 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. More generally, our algorithm works for any class of graphs for which finding disjoint paths between pairs of vertices can be done in polynomial time. Finding disjoint paths between pairs of vertices is NP-hard for directed graphs, even when the number of pairs is only two [17]. However, finding disjoint paths between pairs of vertices is polynomial time solvable in undirected graphs [21], tournament graphs [12], directed planar graphs [25], and directed acyclic graphs [28].

Theorem 4 *Let $\mathcal{C}$ be the class of (di)graphs for which finding disjoint paths between pairs of vertices can be done in polynomial time. Then given a graph $G \in \mathcal{C}$, 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 it 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 [21], find the vertex disjoint paths between the pairs of vertices in S', in

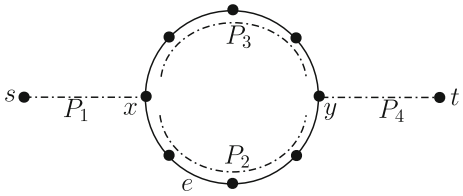

Fig. 9 Cycle without any tracking edges

the graph G'. If G is a tournament graph, then using the algorithm for disjoint paths in tournaments from [12], find the vertex disjoint paths between the pairs of vertices in S', in the graph G'. Since the disjoint path problem can be solved in polynomial time for undirected graphs and tournaments [12,21], 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 exist 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 pairs of vertices in S', then P does not exist. In this case the algorithm returns NO. Since the algorithm relies on finding disjoint paths, apart from chordal and tournament graphs, it is applicable for all graphs for which disjoint paths can be found in polynomial time. □

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 12 *In a reduced graph $G = (V, E)$, a set of edges $T \subseteq E$ is a tracking edge set only if T contains an edge from each cycle in G.*

Proof Suppose the claim does not hold. Then there exists a cycle C in graph G, such that $E(C) \cap T = \emptyset$, i.e. none of the edges in C belong to T. Consider an edge $e \in V(C)$. Due to Reduction Rule 1, e participates in an s-t path, say P. Let x be the first vertex of C that appears in path P while traversing from s to t. Similarly, let y be the last vertex of C that appears in path P while traversing from s to t. See Fig. 9. Observe that x and y serve as local source and sink respectively for the cycle C, and there exist exactly two vertex disjoint paths between x and y in C. Since none of the edges in C are part of the tracking edge set T, this leads to two s-t paths in G with exactly same sequence of edges. This contradicts the fact that T is a tracking edge set for G. □

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 13 *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 $T \cap (E(C) \setminus \{a, b\}) = \emptyset$.*

From Lemmas 12 and 13 we have the following corollary.

Corollary 2 *In a reduced graph G, a set of edges $T \subseteq E(G)$ is a tracking edge set for G if and only if T is a feedback edge set F for G.*

Although finding a minimum FVS is an NP-hard problem, an FES can be found in polynomial time. Hence, we have the following theorem.

Theorem 5 *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. We start with the application of Reduction Rule 1. From Corollary 2 we have that a set of edges is a tracking edge set for G if and only if it is an FES 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 [13]. 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. A constructive algorithm has also been given that helps identify an s-t path, given

the unique sequence of trackers it contains. We also analyze the problem TRACKING PATHS USING EDGES, and prove it to be polynomial time solvable. 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. It would be interesting to explore whether the algorithms for chordal graphs and tournaments can be extended to work for oriented chordal graphs. Open problems also include exploring constant factor approximation algorithms for bounded degree graphs and other NP-hard variants of the problem for both undirected and directed graphs.

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

References

1. Bafna, V., Berman, P., Fujito, T.: A 2-approximation algorithm for the undirected feedback vertex set problem. SIAM J. Discrete Math. **12**(3), 289–297 (1999)
2. 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.) Algorithms and Computations, pp. 142–151. Springer Berlin Heidelberg, Berlin (1995)
3. Banik, A., Katz, M.J., Packer, E., Simakov, M.: Tracking paths. In: Algorithms and Complexity—10th International Conference, CIAC 2017, pp. 67–79 (2017)
4. Banik, A., Choudhary, P.: Fixed-parameter tractable algorithms for tracking set problems. In: Algorithms and Discrete Applied Mathematics—Proceedings of the 4th International Conference, CALDAM 2018, Guwahati, India, February 15–17, 2018, pp. 93–104 (2018)
5. Banik, A., Choudhary, P., Lokshtanov, D., Raman, V., Saurabh, S.: A polynomial sized kernel for tracking paths problem. Algorithmica **82**(1), 41–63 (2020)
6. Banik, A., Choudhary, P., Raman, V., Saurabh, S.: Fixed-parameter tractable algorithms for tracking shortest paths (2020). arXiv:2001.08977
7. Bellitto, T.: Separating codes and traffic monitoring. Theor. Comput. Sci. **717**, 73–85 (2018). Selected papers presented at the 11th International Conference on Algorithmic Aspects of Information and Management (AAIM 2016)
8. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Tracking routes in communication networks. In: Censor-Hillel, K., Flammini, M. (eds.) Structural Information and Communication Complexity, pp. 81–93. Springer, Cham (2019)
9. Charbit, P., Thomassé, S., Yeo, A.: The minimum feedback arc set problem is np-hard for tournaments. Comb. Probab. Comput. **16**(1), 1–4 (2007)
10. Choudhary, P.: Polynomial time algorithms for tracking path problems. In: Combinatorial Algorithms—Proceedings of the 31st International Workshop, IWOCA 2020, Bordeaux, France, June 8–10, 2020, pp. 166–179 (2020)
11. Choudhary, P., Raman, V.: Improved kernels for tracking path problems. CoRR (2020). arXiv:2001.03161
12. Chudnovsky, M., Scott, A., Seymour, P.: Disjoint paths in tournaments. Adv. Math. **270**, 582–597 (2015)
13. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. The MIT Press, Cambridge (2001)

14. 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
15. 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, December 8–11, 2019, Shanghai University of Finance and Economics, Shanghai, China, pp. 54:1–54:17 (2019)
16. 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
17. Fortune, S., Hopcroft, J., Wyllie, J.: The directed subgraph homeomorphism problem. Theoret. Comput. Sci. **10**(2), 111–121 (1980)
18. Garey, M.R., Johnson, D.S., Stockmeyer, L.: Some simplified np-complete graph problems. Theoret. Comput. Sci. **1**(3), 237–267 (1976)
19. Geman, D.: Random fields and inverse problems in imaging. In: Hennequin, P.-L. (ed.) École d'Été de Probabilités de Saint-Flour XVIII, 1988, pp. 115–193. Springer Berlin Heidelberg, Berlin, Heidelberg (1990)
20. Golumbic, M.C.: Chapter 3—perfect graphs. In: Golumbic, M.C. (ed.) Algorithmic Graph Theory and Perfect Graphs, pp. 51–80. Academic Press, London (1980)
21. Kawarabayashi, K., Kobayashi, Y., Reed, B.: The disjoint paths problem in quadratic time. J. Comb. Theory Ser. B **102**(2), 424–435 (2012)
22. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. J. R. Stat. Soc. Ser. B (Methodol.) **50**(2), 157–224 (1988)
23. McGarvey, D.C.: A theorem on the construction of voting paradoxes. Econometrica **21**(4), 608–610 (1953)
24. 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
25. Schrijver, A.: Finding k disjoint paths in a directed planar graph. SIAM J. Comput. **23**(4), 780–788 (1994)
26. Speckenmeyer, E.: On feedback problems in digraphs. In: Nagl, M. (ed.) Graph-Theoretic Concepts in Computer Science, pp. 218–231. Springer Berlin Heidelberg, Berlin (1990)
27. Stearns, R.: The voting problem. Am. Math. Mon. **66**(9), 761–763 (1959)
28. Suurballe, J.W., Tarjan, R.E.: A quick method for finding shortest pairs of disjoint paths. Networks **14**(2), 325–336 (1984)