1
Introduction

One of the commonly studied problems in areas of networks and security is tracking of moving objects. Typically a secure facility or network needs to be monitored for the paths traced by moving entities within. This is usually implemented by placing sensor nodes at various checkpoints and recording movement of entities with respect to the checkpoints. Popular application scenarios are tracking of moving objects in telecommunication networks and road networks. The goal can be efficient and optimized tracking of objects, for the purpose of surveillance, monitoring, intruder detection, and operations management.

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. Consider the example of a security system at a large airport. As a security measure it is required to identify the route taken by passengers across the airport from entry to departure or from arrival to exit. A set of carefully chosen security scan points can be selected as identification points to trace the movement of passengers.

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 can help detect potential network flaws, study traffic patterns of moving objects, optimize network resources based on such patterns, and for other such network analysis based tasks. Tracking has been largely studied in the fields of machine learning, artificial intelligence, networking systems among other fields.

Graphs serve as a systematic model for modeling and analysis of many real life problems. We analyze the problem of tracking by studying a graph theoretic version of the problem. We consider a graph with designated source and destination vertices. The problem asks for a set of vertices that are capable of uniquely identifying each source-destination path. Specifically, the goal is to identify a small set of vertices, referred to as *trackers*, such that the sequence of trackers in each path between the source and the destination is unique. In other words, we refer to this set of vertices/trackers as a *tracking set*.

Definition 1.1. ***Tracking Set****: For a graph $G = (V,E)$ with terminal vertices s,t, a set of vertices (edges) $T \subseteq V$ ($T_e \subseteq E$) is a tracking set if for any simple path P between s and t, the sequence of vertices (edges) from T (T_e) encountered in P is unique.*

The Tracking Paths problem 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 $T \subseteq V$ for G.

Here, terminal vertices s and t are the designated source and destination vertices respectively, and, *tracking set* is the set of vertices (edges) whose intersection with each path between s and t

should result in an unique sequence. As per our definition, a tracking set can be constructed using either vertices or edges. For a major part of this thesis, we focus on finding a tracking set using vertices. Hence, unless otherwise specified, by *trackers* we mean vertices marked as trackers, and by *tracking set*, we mean a tracking set comprising vertices.

It is important to note that in order to distinguish paths in a graph, the sequence of vertices in paths is required to be unique. See Figure 1.1.

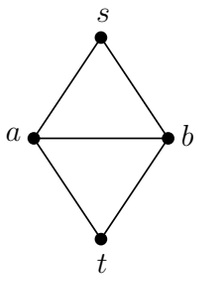

Figure 1.1 : Different sequences of the same vertex set can result in different paths.

Consider a path P_1 consisting of the vertex sequence (s,a,b,t), and a path P_2 consisting of the vertex sequence (s,b,a,t). Observe that both P_1 and P_2 pass through the same set of vertices. However, it is the sequence of vertices they contain, that distinguishes these paths. Hence, while tracking all paths between s and t, we need to ensure that a tracking set leads to a unique sequence of trackers encountered in each of these paths.

Owing to practical scenarios, it might make sense for entities in a network to take only the shortest path(s) between source and destination. If there are multiple shortest paths between source and destination, then the problem can be modified into focusing only on shortest paths in the graph. The problem is formally defined as follows

Tracking Shortest Paths (G,s,t)
Input: An undirected graph $G=(V,E)$ with terminal vertices s and t.
Question: Find a minimum cardinality tracking set $T \subseteq V$ with respect to all shortest paths between s and t in G.

Although we define a tracking set as a set of vertices that ensures a unique sequence of trackers for each s-t path (a simple path between s and t) or shortest s-t path (a shortest path between s and t), in the case of Tracking Shortest Paths it is sufficient for a tracking set to have a unique intersection with the vertex set (rather than the sequence) of each shortest path between s and t. This is due to the fact that if two s-t paths contain the same set of vertices, but in different sequences, at least one of them is not a shortest s-t path. Observe that for a graph G, a tracking set for all s-t paths is also a tracking set for all shortest s-t paths. However, it may be the case that G does not have a tracking set of size at most k for all s-t paths, but it might still have a tracking set of size at most k for all shortest s-t paths. For example, the graph in Figure 1.1 does not have a tracking set for all s-t paths of size $k=1$, but it has a tracking set for shortest paths (can consist of either a or b) of size $k=1$. Hence the parameterized complexity of Tracking Shortest Paths is a problem of independent interest.

These problems can be generalized to a combinatorial setting, wherein groups of elements (shared/repeated among groups) are required to be distinguished using a small set of elements. Formally, we define the problem as follows

TRACKING SET SYSTEM
Input: A set system $\mathcal{P} = \{X, \mathcal{S}\}$.
Question: Find a minimum cardinality tracking set $T \subseteq X$, such that for any two distinct $S_i, S_j \in \mathcal{S}$, it holds that $S_i \cap T \neq S_j \cap T$.

Note that for TRACKING SET SYSTEM a *tracking set* is defined as a subset of elements from the universe, whose intersection with each set in the family is unique. All the three problems described above were found to be NP-hard [8, 10, 45], i.e. under standard complexity theoretic assumptions, it is not expected that these problems will have polynomial time[1] solutions. This makes it infeasible to devise algorithms/programs to solve them. This is not surprising as many problems naturally occurring in the real world, and found to be NP-hard when modeled theoretically. Multiple approaches have been used to tackle NP-hard problems. Some of these are Approximation, Heuristic Methods, Parameterized Analysis, and Randomization. Approximation aims at computing efficient algorithms that give near optimum solutions. Heuristic techniques employ practical methods that need not guarantee an optimal solution but lead to satisfactory results. Parameterized analysis involves multivariate analysis of the problem, by considering parameter(s) other than the input size as well. This serves as a practical approach, since many problems occurring in the real world do not appear in their most general form, and have some pattern or additional characteristics. This additional information can sometimes be captured in form of a parameter and thus be used to find solutions with polynomial running time for the cases when the parameter value is much smaller than the input. Randomized algorithms use random numbers as a part of the logic to decide the next step in their algorithm. In this thesis we mainly study parameterized analysis of the above mentioned problems, specifically TRACKING PATHS and TRACKING SHORTEST PATHS.

1.1 USEFUL DEFINITIONS

In this section we give definitions for some standard terminology relevant to this chapter. A more detailed description on notations and terms used in this thesis can be found in Chapter 2.

NP-hard

A decision problem is NP-hard if every problem in NP (Non-deterministic Polynomial time decidable) can be reduced to it in polynomial time. NP-hard problems are at least as hard as any problem in NP, and a solution to any NP-hard problem would imply a solution for all problems in NP. Currently, there are no known deterministic polynomial time algorithms for NP-hard problems.

1.1.1 Parameterized Complexity

Parameterized complexity is a branch of computational complexity theory that deals with multivariate analysis of a problem which allows a finer classification of the hardness of the problem compared to the classical setting where the problem hardness depends only on the input size. In Parameterized Complexity, we associate a problem with a parameter (an integer), preferably much smaller than the input size, and we analyze the complexity of the problem with respect to both the input size and the parameter.

Fixed-Parameter Tractable

Parameterized problems are always associated with a parameter. A fixed-parameter tractable (FPT) algorithm is the one whose running time can be expressed as $f(k).n^{\mathcal{O}(1)}$, where f is a computable function and k is the parameter. We say that a parameterized problem is fixed-parameter tractable if it admits an FPT algorithm. FPT is also used to denote the class of parameterized

[1] By polynomial time we mean a running time that has polynomial dependence on the input size.

problems that admit fixed-parameter tractable algorithms.

Kernelization

Kernelization involves reducing a parameterized problem into an equivalent instance (referred as *kernel*) in polynomial time, such that the size of the new (reduced) instance is a function of the parameter alone. If the size of the new instance can be bounded by a polynomial function of the parameter, we call it a polynomial kernel. Observe that if a problem admits a kernel, it is FPT as well, since any exponential (in terms of input) time algorithm can be used to solve the problem as the size of the reduced instance is a function of the parameter alone. It is also known that if a problem admits an FPT algorithm, it also admits a kernel.

W-hardness

The W-hierarchy is used to capture the theory of intractability of parameterized problems. The hierarchy is organized as this $FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq XP$, $W[1]$-hard being the least hard among the complexity classes that capture hardness.

1.1.2 Approximation

An approximation algorithm is the one that finds a solution to a problem with provable guarantees on how close the solution is to an optimum solution. Approximation algorithms are usually studied for NP-hard optimization problems. The ratio between the value of an optimum solution and the one obtained from an approximation algorithm is known as the *approximation ratio*.

APX-hard

APX denotes the class of NP optimization problems that admit polynomial time approximation problems with approximation ratio that is bounded by a constant. A problem is said to be APX-hard if there exists an approximation preserving reduction from every problem in APX to it.

More details on Parameterized Complexity and Approximation can be found in Chapter 2.

1.1.3 Problem Definitions

Here we define some terms and problem definitions that are relevant to this chapter.

Vertex Cover

For a graph G with vertex set $V(G)$ and edge set $E(G)$, $S \subseteq V(G)$ is said to be a vertex cover for G, if the graph induced by the vertex set $V(G) \setminus S$ does not have any edges. The Vertex Cover problem requires finding a vertex cover of minimum size.

Feedback Vertex Set

For a graph G with vertex set $V(G)$, $S \subseteq V(G)$ is said to be a feedback vertex set for G, if the graph induced by the vertex set $V(G) \setminus S$ does not have any cycles. The Feedback Vertex Set problem requires finding a feedback vertex set of minimum size.

Hitting Set

For a set system $\mathcal{P} = \{X, \mathcal{F}\}$, a set $H \subseteq X$ is known as a hitting set if $X \cap F \neq \emptyset$ for every $F \in \mathcal{F}$. The Hitting Set problem requires finding a hitting set of minimum size.

Test Cover

Given a set of items $\{1,2,\ldots,n\}$ and a family of subset of those items $\mathcal{M}$, a *test cover* $\mathcal{T} \subseteq \mathcal{M}$ is a collection of sets (also called *tests*) such that for each pair of items $x,y \in [n]$, there exists a set (*test*) $T \in \mathcal{T}$ such that $|T \cap \{x,y\}| = 1$. The TEST COVER problem requires finding a test cover of minimum size.

Since a majority of the results included in this thesis are from parameterized complexity, we now define the parameterized versions of the problems that are relevant to the thesis.

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

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

TRACKING SET SYSTEM $(\mathcal{P} = \{X,\mathcal{S}\},k)$	**Parameter:** k
Input: A set system $\mathcal{P} = \{X,\mathcal{S}\}$ and a non-negative integer k.	
Question: Is there a tracking set T of size at most k for $\mathcal{P}$?	

Observe that while defining the above parameterized problems, we have defined the parameter as the size of the output i.e. the size of a tracking set. However, it is possible that some other graph parameter is chosen as the parameter.

1.2 PREVIOUS WORK

We start the literature survey by referring to some active area of research in terms of tracking moving objects in networking and security related areas. Later we will discuss the theoretical work done with regards to distinguishing paths in a graph. We further move upon to discuss the work done around distinguishing vertices using other structures in a graph.

1.2.1 Tracking Moving Objects

Tracking moving objects in a secure environment is an active area of research. Typically a secure environment is modeled as a network with fixed entry and exit point(s). Monitoring is achieved by placing sensor nodes which monitor the movements of the objects in the network. For a detailed study of field surveillance for the purpose of habitat monitoring, securing buildings, and intruder tracking please refer to [12, 49]. While tracking traces of illegal activities over the Internet, the biggest challenge is to track moving data packets [75, 82]. Tracking of moving objects has been studied in the field of wireless sensor networks. See [13] for a survey of target tracking protocols using wireless sensor networks. Some researchers have studied this with respect to power management of sensors [44]. Coordinated path tracking and framework for multi-target tracking have been discussed in [87] and [83]. Tracking algorithms can be used in designing debugging tools. Another useful application is the problem of leakage detection systems. In these kinds of problems it would be resource efficient if a small subset of nodes/checkpoints are sufficient to trace the movements of entities in the network. Despite being an active area of research, a major part of this research so far has been based on heuristics.

1.2.2 Tracking Paths in Graphs

In [10], Banik *et al.* formalized the problem of tracking in networks as a graph theoretic problem and did a systematic study. The authors proved that the problem of finding a minimum-cardinality tracking set with respect to the *shortest s-t* paths (Tracking Shortest Paths) is NP-hard and APX-hard. Other results include a heuristic analysis for computing a small tracking set, and a 2-approximation algorithm for solving Tracking Shortest Paths in planar graphs. The authors also give an exact polynomial time algorithm to find a minimum set of trackers such that a shortest source-destination path passes through a designated (received as part of input) forbidden set of vertices if and only if it passed through a tracker. Finally, they also describe how to preprocess a graph and its tracking set, such that given a sequence of visited trackers, one can reconstruct the traversed path P in $\mathcal{O}(|P|)$ time.

Eppstein *et al.* studied Tracking Paths in planar graphs [35]. They proved that the problem remains NP-hard in planar graphs and gave a 4-approximation algorithm for the same. They also used Courcelle's theorem to prove that Tracking Paths can be solved in linear time for bounded clique-width graphs if the clique decomposition is given as a part of the input.

Bilò *et al.* studied Tracking Shortest Paths for both single source-destination and multiple source-destination cases [15]. Here they give an $\mathcal{O}(\sqrt{n})$-approximation algorithm for single source-destination Tracking Shortest Paths and extend it to an $\mathcal{O}(\sqrt{n\log n})$-approximation algorithm for the case when there are multiple sources and destinations. They also prove that Tracking Shortest Paths is NP-hard for multiple source-destination case even in cubic planar graphs. Further they give an $\mathcal{O}(2^{h^2})$ FPT algorithm for the single source-destination case where h is the maximum number of vertices equidistant from the source (or destination).

All the results on Tracking Paths and Tracking Shortest Paths prior to this thesis have been listed in Table 1.1.

Problem	**Complexity**	**FPT**	**Approximation**
Tracking Set System	NP-hard	-	-
Tracking Shortest Paths: Single source-destination			
General graphs	NP-hard	$\mathcal{O}^*(2^{h^2})$	APX-hard,$\tilde{\mathcal{O}}(\sqrt{n}).OPT$
Planar graphs	-	-	$2.OPT$
Planar cubic graphs	NP-hard	-	-
Tracking Shortest Paths: Multiple source-destination			
General graphs	NP-hard	-	$\mathcal{O}(\sqrt{n\log n}).OPT$
Planar cubic graphs	NP-hard	-	-
Tracking Paths: Single source-destination			
Planar graphs	NP-hard	-	$4.OPT$
Bounded clique-width graphs	P	$\times$	$\times$

Table 1.1 : Known Results. For the FPT result, the parameter h is the maximum number of vertices equidistant from the source (or destination). n refers to the number of vertices in the input graph. Results marked with - were not known prior to this thesis/our work, and those marked with $\times$ either do not hold any value or are not expected under standard complexity theoretic assumptions. $\tilde{\mathcal{O}}$ notation ignores the logarithmic factors.

1.2.3 Distinguishing Vertices and Edges in Graphs

Different structural properties of graphs have been studied previously to analyze navigational models in network settings. In a seminal paper in 1975, Slater [79] introduced the concept of metric

dimension of a graph. In graph theory, the metric dimension of a graph G is the minimum cardinality of a subset S of vertices such that all other vertices are uniquely determined by their distances to the vertices in S [52]. One application of metric dimension is the problem of determining the location of an object in a network, depending on its distance from different landmarks in the network [61]. For a survey of metric dimension in graphs see [55].

There has been a lot of work on distinguishing vertices in a graph using other vertices in the graph. Early references include work on *locating-dominating sets* in acyclic graphs, series-parallel network, and fault-tolerant solutions in [80], [24] and [81] Some closely related graph theoretic problems are DISCRIMINATING CODE [20] and IDENTIFYING CODES [16], [59], [71], where authors have studied identifying vertices using their neighbourhoods. For a graph G, a set of vertices $C \subseteq V$ is said to be a *locating-dominating set* if all vertices in $V \setminus C$ can be uniquely identified by neighbourhood of some vertices in C, and C is said to be an *identifying-code* is all the vertices in V can be uniquely identified by neighbourhood of some vertices in C. Both Locating-dominating sets and Identifying codes have been studied in detail [4], [39], [40], [41], [38]. DISTINGUISHING TRANSVERSAL restricted to neighborhood hypergraphs of graphs is equivalent to problem IDENTIFYING CODES in graphs [16], [59]. DISTINGUISHING TRANSVERSALS when restricted to 2-uniform hypergraphs is equivalent to IDENTIFYING VERTEX COVER, which is the problem of finding a set of vertices $V' \subseteq V$ for a graph $G = (V, E)$, such that $a \cap V' \neq b \cap V'$, for a pair of distinct edges $a, b \in E$. Henning and Yeo give some bounds for the size of an output in [53] and [54] for IDENTIFYING VERTEX COVER and DISTINGUISHING TRANSVERSAL, respectively. There has also been some work on distinguishing edges using paths in a graph [36], [5]. In [43], [42], Foucaud and Kovse have studied the problem of *Identifying Path Cover* which requires finding a set of paths that cover all the vertices in a graph, while uniquely identifying each vertex by inclusion in a distinct set of paths.

While covering problems have received a lot of attention in theoretical computer science, there has been very little work on distinguishing problems. Covering problems usually require covering/hitting some specific set of objects or structures in the input. While distinguishing problems aim at distinguishing among some specific objects or structures. This thesis serves as a work on this less focused upon field of study, while leaving with a lot of related open problems that could serve a lot of practical importance.

1.3 OUR RESULTS

In this section we summarize the results included in this thesis.

1.3.1 Tracking Set Systems

Given a general set system $\mathcal{P} = \{X, \mathcal{F}\}$, we can derive a kernel of size $\mathcal{O}(2^{2^k})$ and an FPT algorithm running in time $\mathcal{O}^*(2^{k2^k})$, where k is the parameter and the size of desired tracking set. There exists an $\log f.OPT$ approximation algorithm for TRACKING SET SYSTEM, where f is the maximum frequency of occurrence of elements of X in the sets of $\mathcal{F}$.

We prove that d-TRACKING SET SYSTEM, the restricted case of TRACKING SET SYSTEM where the set sizes in the family are restricted to some constant d is NP-hard. d-TRACKING SET SYSTEM admits a kernel polynomial in k and an FPT with running time $\mathcal{O}^*(c^k)$, where k is the size of the desired tracking set and c is a function of d.

1.3.2 Tracking Shortest Paths

TRACKING SHORTEST PATHS admits a kernel of size $\mathcal{O}(2^k)$ and an FPT algorithm running in time $\mathcal{O}^*(2^{k^2+3k})$, where k is the size of the desired tracking set.

We prove that d-Tracking Shortest Paths, the restricted case of Tracking Shortest Paths where the diameter of the input graph is restricted to a constant d is NP-hard. d-Tracking Set System admits a kernel polynomial in k and an FPT with running time $\mathcal{O}^*(c^k)$, where k is the size of the desired tracking set and c is a function of d.

1.3.3 Tracking Paths

Undirected Graphs

We show that Tracking Paths is NP-hard and polynomial time verifiable, hence NP-complete. We further prove that Tracking Paths admits an $\mathcal{O}(k^2)$ kernel and an FPT with running time $\mathcal{O}^*(k^{2k})$ when parameterized by the size of output, the tracking set. We also show that Tracking Paths admits an $\mathcal{O}(OPT)$-approximate algorithm.

When parameterized by the size of vertex cover or the size of cluster vertex deletion set, we show that Tracking Paths admits an FPT running in time $2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$. For planar graphs, we show that Tracking Paths admits an $\mathcal{O}(k)$ kernel and FPT running in time $\mathcal{O}^*(k^k)$. We also show that finding a tracking set of size $n-k$ is W[1]-hard where n is the number of vertices in the graph and k is the parameter.

For graphs with bounded maximum degree δ, we prove that Tracking Paths remains NP-hard and we give a $2(\delta+1)$ approximate algorithm for the same. We show that Tracking Paths is polynomial time solvable in chordal graphs. Finally we show that Tracking Paths is polynomial time solvable in general graphs if edges are chosen as trackers instead of vertices.

Directed Graphs

Since Tracking Paths is NP-hard for directed acyclic graphs, it is implied that Tracking Paths is NP-hard for general directed graphs as well. We show that Tracking Paths in DAGs admits a kernel of size $\mathcal{O}(2^k)$ and an FPT running in time $\mathcal{O}^*(2^{k^2+3k})$.

Table 1.2 shows a summary of results in this thesis.

1.4 ORGANIZATION OF THE THESIS

The thesis is divided into seven chapters.

Chapter 1 covers the introduction to the problem, motivation and previous work.

In Chapter 2, we give the notations and reference for commonly used terms in the thesis. Here we also give some basic observations and preprocessing techniques for the problem.

Chapter 3 provides analysis for Tracking Set System, Tracking Shortest Paths and Tracking Paths in DAGs. We show that Tracking Set System is FPT as it is related to the well known Test Cover problem. The results for Tracking Set System can be used to solve Tracking Shortest Paths as well, since Tracking Shortest Paths is reducible to Tracking Set System/. Next, we show that Tracking Paths in DAGs is reducible from Tracking Shortest Paths, and we give a better FPT algorithm for Tracking Shortest Paths that applied for Tracking Paths in DAGs as well. We also provide improved FPT results for the variants where the set sizes in the family (for Tracking Set System) and diameter of the graph (for Tracking Shortest Paths) is restricted to a constant d.

Problem	Complexity	Kernel	FPT	Approxi -mation	Section
TRACKING SET SYSTEM					
General Case		$\mathcal{O}(2^{2^k})$	$\mathcal{O}^*(2^{k2^k})$	$(\log f).OPT$	3.2.2
d-TRACKING SET	NP-hard	$k^{\mathcal{O}(1)}$	$\mathcal{O}^*(c^k)$	-	3.2.1
TRACKING SHORTEST PATHS					
General Case		$\mathcal{O}(2^k)$	$\mathcal{O}^*(2^{k^2+3k})$	-	3.3.2
Diameter d graphs	NP-hard	$k^{\mathcal{O}(1)}$	$\mathcal{O}^*(c^k)$	-	3.3.1
TRACKING PATHS in Undirected Graphs					
General Case	NP-hard	$\mathcal{O}(k^2)$	$\mathcal{O}^*(k^{2k})$	$\mathcal{O}(OPT)$	6.3,6.4
$k = \lvert$Vertex Cover$\rvert$	-	-	$2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$	-	7.3
$k = \lvert$Cluster Vertex Deletion Set$\rvert$	-	-	$2^{\mathcal{O}(k^2)}n^{\mathcal{O}(1)}$	-	7.4
Planar Graphs		$\mathcal{O}(k)$	$\mathcal{O}^*(k^k)$	-	6.5
Graphs with max. deg.$=\delta \geq 6$	NP-hard	-	-	$2(\delta + 1)$ $.OPT$	8.4
Chordal Graphs	P				8.3
Tracking set of size $n-k$	W[1]-hard			-	6.6
TRACKING PATHS using Edges	P				8.6
TRACKING PATHS in Directed Graphs					
Directed Acyclic Graphs	NP-hard	$\mathcal{O}(2^k)$	$\mathcal{O}^*(2^{k^2+3k})$	APX-hard, $\tilde{\mathcal{O}}(\sqrt{n}).OPT$	3.3.2

Table 1.2 : Our Results. Unless otherwise specified for FPT results, the parameter k is the size of a tracking set. All tracking paths related results were derived for the single source-destination case. Note that the hardness results shall extend to the multiple source-destination cases as well. f refers to the upper bound on the number of sets in the family in which each element appears in a set system. n refers to the number of vertices in the input graph. $\mathcal{O}^*$ notation ignores the polynomial factors in terms of the size of input. $\tilde{\mathcal{O}}$ notation ignores the logarithmic factors.

Chapter 4 onwards we start the analysis of TRACKING PATHS, wherein the goal is to find a set of trackers that can distinguish between all s-t paths. Chapter 4 provides an NP-hardness, through reduction from VERTEX COVER, along with giving a polynomial time verification algorithm thus proving the problem NP-complete. Next, we give a polynomial kernel for the problem when the parameter is the size of the output, using basic preprocessing and counting techniques. We then give a lower bound for the number of trackers required in a specific graph structure, which we refer to as the *tree-sink structure*. Using this we improve the kernel to a quadratic kernel. We further give a simple kernelization algorithm for the undirected planar graphs. The chapter also shows that finding a tracking set of size $n-k$ is W[1]-hard, where n is the number of vertices in the input graph and k is the parameter.

Chapter 5 studies structural parameterization for TRACKING PATHS. Since the problem is trivially solvable for edgeless graphs and cliques, it is interesting to study graphs that are k vertices away from such graphs. We show that TRACKING PATHS is FPT when parameterized by the size vertex cover and cluster vertex deletion set for the graph. These are derived using the already known set of preprocessing techniques and some careful case analysis of some specific small subgraphs that can appear in the input.

Chapter 6 considers some restricted cases of Tracking Paths and gives polynomial time algorithms for the same. It is shown that Tracking Paths is polynomial time solvable for chordal graphs.The chapter also gives an approximation algorithm for the case when the maximum degree of graph is bounded by a constant. It is further proven that the problem of Tracking Paths is polynomial time solvable if edges are chosen as trackers instead of vertices. This is done using the relation between Tracking Paths and Feedback Vertex set. We also give a simple polynomial time algorithm, which if given a tracking set for a graph, and a sequence of trackers, can generate the unique path corresponding to that sequence of trackers, if one exists.

Chapter 7 concludes the thesis with a summary of the work done and some discussion on future scope.

...