Continuous Spatio Temporal Tracking of Mobile Targets

by

Harsh Vachhani

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved November 2014 by the
Graduate Supervisory Committee:

Arunabha Sen, Chair
Srikanth Saripalli
Shahrzad Shirazipourazad

ARIZONA STATE UNIVERSITY

December 2014

ABSTRACT

There has been extensive study of the target tracking problems in the recent years. Very little work has been done in the problem of continuous monitoring of all the mobile targets using the fewest number of mobile trackers, when the trajectories of all the targets are known in advance. Almost all the existing research discretized time (and/or space), or assume infinite tracker velocity. In this thesis, I consider the problem of covering (tracking) target nodes using a network of Unmanned Airborne Vehicles (UAV's) for the entire period of observation by adding the constraint of fixed velocity on the trackers and observing the targets in continuous time and space. I also show that the problem is NP-complete and provide algorithms for handling cases when targets are static and dynamic.

DEDICATION

To my family

ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my advisor Dr. Arunabha Sen for his continuous support of my study and research, for his patience and immense knowledge. Thanks to him, I had the opportunity to work on many research topics. I appreciate all his contributions including his time and patience, which helped me throughout the time of my masters and writing of this thesis. He taught me, both consciously and unconsciously, to learn from myself and how the research is done. He has supported me not only financially but also gave me support and the freedom I needed.

For this thesis, I would also like to thank my committee members Dr. Srikanth Saripalli and Dr. Shahrzad Shirazipourazad who provided encouraging and constructive feedback. I am grateful for their time, interest and helpful comments on my work. I praise the enormous amount of help and support of Dr. Shirazipourazad who guided me from the beginning of my thesis. She was the first one with whom I worked in our lab and then she later became a mentor and a friend. Her scientific advice and suggestions were instrumental for my research work. She always supported me and I would have never been able to finish without her guidance.

The members of my lab have contributed immensely to my personal and professional time at ASU. The group has been source of friendships and as well as good advice. I am especially grateful to previous lab member Dr. Sujogya Banerjee, who had provided some good advice when I joined the research lab. I thank my current fellow lab-mates Arun Das (provided helpful guidance, moral support and teaching me how to approach problems), Anisha Mazumder (friendly, nice and smart girl who I'm glad to be friends with), Joydeep Banerjee (always friendly and cheerful) and Zahra Derakhshandeh (encouraging and helpful friend), for the stimulating discus-

sions and for the time and effort they spent in the research work related to my thesis. It would have been lonely in the lab without them. I would like to acknowledge Brian Bogard who worked with me on few projects. I was always amazed by his speed of writing code and his ability to manage time. I very much appreciate him for giving me a great opportunity to work with him after my studies.

Dozens of people have helped me immensely here at ASU. I have appreciated and gratefully acknowledged the help and support by Professor Mutsumi Nakamura and Professor Alan Skousen. They provided financial assistance and I enjoyed taking their courses. I also thank Daniel and Rinkal for the opportunity to intern at Dow Agrosciences. I had a great time with other interns and I learned a lot during those three months. Thank you to everyone from 5th floor BYENG staff, CIDSE Advising, and ISSC (especially Monica Dugan who has been an amazing help all this time).

Also I thank my friends in Arizona State University. My life here was made enjoyable in large part due to many friends that became part of my life. I am grateful for the time spent with my roommate and good friend Pankaj Khatkar, for the hiking trips, intellectual debates and his encouragement. He taught me cooking and many useful life skills. Thanks for being such a great friend. I also thank Khushboo(and her family), Himanshu, Sudip, Rashmin, Urvish, Anindita, Parminder, Dipal, and Shreeniwas for giving me so many good memories. I value their friendship and support. I also thank Mr. Bhagirath Gohil, Mr. Nilesh Patel and their families who always welcomed me to their home and invited me over during holidays.

I have special friends to thank, going back to pre-ASU days for their continued friendship since my undergrad. Jyoti and Sourabh, thanks for being there to listen, bear the frustrations and share the joy of successes during this journey. I am grateful for all the memorable trips and your support. Ateendra and Shashank, thanks for influencing me in creative pursuits and inspiring me. I also thank my friends

Tanupriya, Divya, KV, Mustahsan, Kunal, Priyesh, Prakhar and Amol for providing support and friendship that I needed. I have to thank a special teacher and manager, Mrs. Shipra Mudgal and Mr. Mahendra Nath, for their help when I was trying to apply for higher studies and also for their reference.

Lastly, I would like to thank my family: My parents, Rajesh Vachhani and Munila Vachhani, for their infinite love and for supporting me in all my pursuits. My brother Shilp, who has been my oldest best friend. My Nani, who supports and loves me. Thank you.

TABLE OF CONTENTS

Page

LIST OF FIGURES viii

CHAPTER

1 INTRODUCTION 1

1.1 Background 1

1.2 Motivation 3

1.3 Related Work 4

1.4 Overview 5

2 NETWORK MOBILITY MODEL 6

2.1 Mobility Models 6

2.1.1 Random Walk Mobility Model 7

2.1.2 Random Waypoint Mobility Model 7

2.1.3 Random Direction Mobility Model 8

2.1.4 Pursue Mobility Model 8

2.1.5 Exponential Correlated Random Mobility Model 9

2.1.6 Column Mobility Model 9

2.1.7 Nomadic Community Mobility Model 9

2.2 Our Model Description 10

3 PROBLEM FORMULATION 12

3.1 Preliminaries 12

3.2 Restrictions/Assumptions 12

3.3 Problem Statement 14

3.4 Geometric Disc Cover Problems 15

3.5 NP Completeness Proof of our Problem 16

4 ALGORITHM FOR STATIC CASE 18

CHAPTER Page

4.1 SCR Algorithm 18

4.2 Algorithm for Static Targets 20

4.2.1 Description 20

4.2.2 Maximum Matching in a Bipartite Graph 21

4.2.3 Network Max Flow Problem 21

4.2.4 Ford Fulkerson Algorithm 22

4.3 Graph Construction 23

4.4 Algorithm PseudoCode 25

4.5 Analysis 25

4.6 Alternate Approaches 27

5 ALGORITHM FOR DYNAMIC CASE 29

5.1 Description 29

5.1.1 Min-Heap Data Structure 30

5.2 Events 30

5.3 Propagation 32

5.4 Algorithm PseudoCode 33

6 VISUALIZATION 37

6.1 Visualization Software 37

6.1.1 Back-end 37

6.1.2 Front-end 39

6.1.3 Interface 39

7 CONCLUDING REMARKS AND FUTURE WORK 41

REFERENCES 44

LIST OF FIGURES

Figure		Page
1.1	Uavs Track a Target in Different Environments	1
3.1	Rectangle Encapsulated Cover in Disk Model for Tracker	13
4.1	An Example of a Region with Target Observation Area and Tracker Base Stations	19
4.2	Covering of Static Targets at Time t_1	24
4.3	Trackers Selected for Covering of Targets at Time t_1	24
4.4	Multiple Solutions for Covering of Targets at Time t_1	28
5.1	Case 1 for Event in Dynamic Cover	31
5.2	Case 2 for Event in Dynamic Cover	32
5.3	Propagation Example	33
6.1	Software Interface Screenshot	40
7.1	Covering by rectangles	42
7.2	Covering by disc	43

Chapter 1

INTRODUCTION

1.1 Background

There has been dramatic growing research in the field of airborne networks and its military and civilian applications. During the last years, an increased use of UAV's (Unmanned Airborne Vehicle) has been noticed. The invention of light materials, low energy consumption machines and high performance processing units led to the construction of flexible flying drones (Chen *et al.* (2009)). Figure 1.1 (image source: http://jaredzichek.com/) shows a UAV tracking two ships.

Figure 1.1: Uavs Track a Target in Different Environments

Supported by the advances in sensing and wireless communication technologies, Airborne networks hold promise in providing practical, wide-applicable, low-cost, and secure information exchange among airborne vehicles. The ability for multiple UAVs

to organize and cooperate for the tracking of a moving target (SUV, vessel, tank, etc.) has prime benefit for intelligence operations for tracking evading targets with minimal interaction from operators. This means that one operator could control many UAVs to monitor many targets, thereby gaining more time. These teams of vehicles open up very important research questions such as: What information is shared between the vehicles? How do the vehicles cooperatively plan trajectories and tasks for each mission type? How can the impact of communications constraints, such as outages and dropped packets be minimized?

UAVs are being used increasingly for target tracking problems. The use of UAVs has gained considerable momentum given their success in recent military operations, and their promise for important domestic applications such as border and coast patrol, fire perimeter monitoring, search and rescue, etc. The UAVs can be used to accomplish tasks that are dangerous for human operators, and they can reduce operational costs by implementing tasks using smaller and cheaper UAVs rather than larger more expensive ones (Zhan *et al.* (2005)).

Typically, target tracking involves two steps. First, it needs to estimate or predict target positions from noisy sensor data measurements by autonomous tracking of moving and evading targets. Second, it needs to control mobile sensor tracker to follow or capture the moving target. For both of the steps the tracker needs to report to a centralized database (without operator attention): the position, position history, velocity vector.

Reporting expected position of the assigned target enables Intelligence, Surveillance, and Reconnaissance (ISR) missions to be more cost effective and efficient. ISR resources have generally been regarded as the number one resource scarcity during recent operations in Iraq and Afghanistan. ISR resources will continue to be limited by personnel constraints unless the vehicle's autonomy and cooperation can be

increased (Wheeler *et al.* (2006)).

The following scenario provides an example motivation for this work: A ground vehicle moves at high speed along a hill road, away from a town where it committed a hit and run attack on a group of civilians. The security forces want to find the attackers and follow them to their hideout. Multiple UAVs are sent to the area where the attackers were reported to have retreated. One of the UAVs identifies the attacker's vehicle based on reports from the civilians and notifies nearby UAVs of its location. The UAVs alter their flight paths relative to that point, to look for other attacker vehicles or by maintaining position near it, without attracting attention from the attackers. As the attackers move through the region, responsibility for direct sensing passes from one UAV to another to always keep the target in sight. Through collaboration, autonomous unmanned air vehicles would be able to complete tasks that each could not have done alone.

In the work mentioned here, we focus on the problem of continuous tracking of all mobile targets using the fewest number of mobile trackers, when the trajectories of all the targets are known in advance. This optimization where the number of trackers are minimized to reduce costs and to use the trackers efficiently has received very little attention. Although it may appear that the assumption regarding complete knowledge of the trajectories of the target makes the problem simple, we show that this problem remains hard, i.e. NP-complete.

1.2 Motivation

Most of the previous work is (i) too impractical to implement(running time), (ii) assume the trackers have infinite speed, or (iii) cannot guarantee cover for continuous time and space. Motivated by the importance of target tracking in military and civilian environment, we attempt to develop algorithms that does not fall in any of

the above cases and are easily implementable.

1.3 Related Work

Considerable research has been conducted in target tracking using UAVs and mobile sensors (Zorbas *et al.* (2013); Zhan *et al.* (2005); Wheeler *et al.* (2006); Nitinawarat *et al.* (2011); Xu *et al.* (2013); Zou and Chakrabarty (2007)). All the fundamental tools of how a target can be detected, how to takeoff and how to land a drone are well analyzed in the literature (Fujita and Shimada (2007); Hsia *et al.* (2010); Kubota and Iwatani (2011)). There also has been large body of research on target tracking using sensor networks. The authors in (Naderan *et al.* (2009)) provide a survey of these studies. Though, most of these studies have static sensor nodes and the concept of path planning of trackers does not exist.

Generally, the target tracking problem using mobile trackers has two elements- (i) estimating target positions using sensor(tracker) data and (ii) managing mobility of trackers. Many of the studies on target tracking using mobile trackers focus on the quality of detection of mobile targets with a given set of UAV's. The authors in (Zou and Chakrabarty (2007)) track a single target using distributed mobility management of a given set of mobile sensors. In (), authors study a case when there is one mobile tracker, a set of static sensors and a mobile sensor controller. The controller receive data from the tracker and sensors and estimate the location of the target. In (Adamey and Ozguner (2012)) the authors study multi target tracking using multiple UAVs and develop a decentralized approach for target location estimation and UAV mobility management. The authors in () focus on energy efficiency issues, related to mobile target tracking and they give a power efficient solution by adjusting the UAVs altitude. In spite of the extensive studies, the problem of finding minimum mobile trackers during the continuous time tracking of the target has been studied by very

few.

In (Zorbas *et al.* (2013)), the authors study a similar problem in which their goal is to find the smallest set of mobile backbone trackers such that the regular trackers are always under coverage of at least one backbone tracker. The work in (Zorbas *et al.* (2013); Srinivas *et al.* (2009)) assume that the trackers have infinite speed. In (Radhakrishnan and Saripalli (2010)), the authors find the optimal set of trackers, consider finite speed of tracker, and compute the solution of coverage problem for each discrete time instance using a greedy algorithm. They also propose a motion assignment algorithm determining the motion of UAVs. The greedy algorithm proposed does not guarantee optimality of the solution.

1.4 Overview

The thesis is organized as follows. In Chapter 2, I describe some of the mobility models and propose our own mobility model for target tracking. Chapter 3 presents the problem formulation, assumptions and proof of NP-completeness for our problem. In Chapter 4, I provide the extension of existing algorithm to handle the covering of static targets and provide theoretical analysis. Chapter 5 contains the detailed description of proposed algorithm for continuous tracking of mobile targets. In Chapter 6, the software for simulation and calculations of the proposed model is discussed. Finally, a brief conclusion and discussion about future work is provided in Chapter 7.

Chapter 2

NETWORK MOBILITY MODEL

2.1 Mobility Models

The mobility models commonly serve as the fundamental mathematical frameworks for network connectivity studies, network performance evaluation, and the design of reliable routing protocols (Ravikiran and Singh (2004)). It is imperative that mobility models are able to accurately capture the movement pattern of each mobile node, based on which information related to the varying network configuration and structure can be derived. Information such as node distribution, movements, link and path should be simulated using such models.

There has been many mobility models proposed in the recent research literature (Bai and Helmy (2004)). They are generally classified into two types: traces and synthetic models. Traces are mobility patterns which are observed in real life systems, and provide accurate information, especially when they involve appropriately long observation period and large number of participants (Camp *et al.* (2002)). However, some network environments like ad-hoc networks are not easily modeled if traces are not available. In this situation, it becomes necessary to use synthetic models which attempt to realistically represent the behaviors of MNs.

There are many entity and synthetic mobility models in literature. Out of them I will briefly discuss interesting ones such as Random Walk(Hong *et al.* (1999)), Random Waypoint(Johnson and Maltz (1996)), Random Direction(Gloss *et al.* (2005)), which are the entity models and Pursue, Exponential Correlated Random, Column, Nomadic Community Mobility Model(Camp *et al.* (2002)), which are the synthetic

models. Some of the mobility models have been studied extensively, the most well known being random direction (RD), and random waypoint (RWP) (Bettstetter *et al.* (2004); Hyytia *et al.* (2006); Le Boudec and Vojnovic (2005); Yoon *et al.* (2003b)). The random walk(RW) is also a widely used model (Bar-Noy *et al.* (1995); Garcia-Luna-Aceves and Madruga (1999); Rubin and Choi (1997); Zonoozi and Dassanayake (1997)).

We believe that mobility models need to be application-specific, due to the wide range of variability in their applications, and different movement patterns associated with each application. Therefore it is worthwhile to propose our own synthetic model by extending one of the popular mobility models, by focusing on: a) the specific application, and b) the movement pattern associated with our work.

2.1.1 Random Walk Mobility Model

The Random Walk Mobility model was designed for mimic unpredictable movements that is observed in nature. In this model, an MN moves from its current position to new position by selecting random speed and direction to travel. The speed and direction are taken from predefined ranges. Each movement occurs either for constant time interval t or till constant distance is traveled, after which new speed and direction is computed. It is a memory less model and each movement's speed or direction is independent of the previous movement speed or direction. If the MN encounters a border, it bounces off depending on the angle of incoming direction.

2.1.2 Random Waypoint Mobility Model

The RWP model assumes that each node chooses a random destination (waypoint) and traveling velocity; upon the arrival, it pauses for specified time before traveling to the next waypoint. The mobile nodes(MNs) are initially distributed randomly

around the simulation area. It includes pause times between changes in direction and/or speed. The node begins by staying in one random location for a certain time period. Once the time expires, the node chooses a random destination(in the simulation region) and speed(uniformly distributed between $(V_{min}, V_{max}]$, where V_{min} and V_{max} are the min speed and max-speed respectively). The node then travels to the new location with the chosen speed and once it reaches the destination, it pauses for specific time again. This process repeats for the entire simulation time.

2.1.3 Random Direction Mobility Model

RD models assume that nodes choose random direction and then travels to the region boundary. After it has reached the border, it will pause for a fixed time and then choose a direction from a range to travel to another boundary waypoint, continuing the process.

An extended version of it assumes that each node randomly chooses a speed and direction after the completion of a randomly selected traveling time and no longer needs to touch the boundary.

2.1.4 Pursue Mobility Model

This model attempts to represent a MN tracking a moving target. The positions are calculated using an update equation $newposition = oldposition + acceleration(targetoldposition) + randomvector$ where random vector is a offset for each node. The value of this vector can be obtained from any entity mobility model. An example for this model could be police cars chasing a criminal.

2.1.5 Exponential Correlated Random Mobility Model

In this model a function is used to create MN motion. Suppose $\overrightarrow{b}(t)$ is the given position at time t, the next position at t+1 (for MN or group) is given as

$$b(t+1) = b(t)\varrho^{-\frac{1}{\tau}} + (\sigma\sqrt{1-(\varrho^{-\frac{1}{\tau}})^2}r$$

where r is a random Gaussian variable with variance σ and τ adjusts the rate of change of the node's previous location to its new location (small τ equates to large change). Unfortunately, it is very hard to create a given motion pattern by selecting appropriate values for τ and σ.

2.1.6 Column Mobility Model

This model represents a set of nodes that mode in a column(or a given line), which is moving in forward direction. For example: a group of young children walking in straight line or a row of soldiers marching towards the enemy. To implement this model, an initial reference column is defined, and a node is placed in relation to a reference point in its reference column. The node then moves around randomly around the reference point(based on predefined offset) using a Random Walk Mobility model or any other Entity model. The new reference point is given by $newreferencepoint = oldreferencepoint + advancevector$ where old reference point is the node's previous reference and advance vector is the offset that moves the column.

2.1.7 Nomadic Community Mobility Model

In this model, each group of MN's have their own personal spaces where they move randomly. Each node uses a entity mobility model for roaming around their reference point. After a random interval of time the reference point changes and the nodes

in the group travel to new location where they begin roaming around new reference point. For example, this model can represent class of students touring a museum.

2.2 Our Model Description

A mobility model should be able to mimic the moments of nodes(targets and trackers). We create a synthetic model based on Random Waypoint for capturing the path of the vehicles. Random Waypoint model(RWP) is a widely used mobility model (Garcia-Luna-Aceves and Spohn (1999); Broch *et al.* (1998); Chiang and Gerla (1998); Johansson *et al.* (1999)) for ad-hoc networks.

We create a simplified version of RWP where we ignore the pause time and change in velocity. In our analysis, the target node trajectory is provided beforehand. The trajectory is piece wise linear function that contains line segments from one waypoint to another. We consider trajectory for node i denoted as $\Pi_i = \{w_1, w_2, \ldots, w_n\}$ is defined as set of n waypoints in a plane, where the starting position is w_1 and final destination is w_n. Each waypoint w_i is denoted by $x - y$ tuple $(x_i, y_i) \forall i$. The node moves from waypoint $w_{i-1}$to w_i in a straight line with constant velocity. All kinds of path can be captured by this model(for e.g. a curve path can be approximated by using series of broken line segments). The model can be used to represent mobile nodes whose movements are independent of each other (targets) and also represent mobile nodes whose movements are dependent (trackers). For representing the mobile nodes with dependent movements, the trajectory waypoints are dependent on the information of the target nodes being pursued.

Our model is robust in capturing the movement of nodes in ad-hoc network when trajectory is known. The most common problem with simulation studies using RWP is a poor choice of velocity distribution (Yoon *et al.* (2003a)) , like having a uniform distribution $U(0, V_{max})$ that leads to average velocity decreasing over time and

reaching a steady state where nodes stop moving. However, we avoid this problem by choosing constant velocity for both targets and trackers. As I discuss in Chapter 2.2, this model does not reflect the the kinematics of turning objects, node acceleration and changes in elevation(of the tracker). It is easy to argue that this model is elementary and may not realistically capture the motion of high-speed airborne vehicles(trackers). Yet it is a robust model, analyzable for node distribution and connectivity analysis. It also provides reasonable performance with a wide range of motion patterns.

Chapter 3

PROBLEM FORMULATION

3.1 Preliminaries

Throughout this paper, we assume that the input consists of set of targets $A = \{a_1, a_2, \ldots, a_n\}$ and their trajectories. Motion is assumed to be in Euclidean 2-dimensional space. The motion of target a_i is entirely described by the set of waypoints $\{w_1, w_2, \ldots, w_z\}$ and the velocity $V = \{v_1, v_2, \ldots, v_{z-1}\}$ such that the velocity is v_i for moving from w_i to w_{i_1}. The initial position of a_i, is w_1 at time t_1. Observe that every target is moving on a straight line between waypoints with the specified velocity.

The input also consists of set of tracker base station $L = \{l_1, l_2, \ldots, l_m\}$ and trackers $B = \{b_1, b_2, \ldots, b_n\}$ available at each base station. The max velocity V_{max} for tracker is provided. The trackers starting position is at their corresponding base station at time t_0. Lead time given by $|t_1 - t_0|$ is the spare time available to trackers to prepare themselves.

We use $dist(i, j)$ to denote distance between any node i and j or their geometric points p_i and p_j respectively.

The proposed algorithm solution will tell us the minimum trackers needed and trajectory(along with velocity) of the each needed tracker.

3.2 Restrictions/Assumptions

Each tracker is assumed to have a sensor (E.g. camera) to detect mobile events. The sensor has a maximum range beyond which it cannot detect any nodes. We

assume the communication channel to be disk connectivity model. The authors in (Srinivas *et al.* (2009)) used the rectangle encapsulated in disks, without any additional modification to obtain an algorithm(MOAC) with 3-approximation ratio for the geometric disk cover problem within the strip. We will also use rectangle encapsulated in disks for our heuristic (Please see Figure 3.1). We consider all the targets within the rectangle area as coverable by the respective tracker. The height H and width W of the rectangle can be changed. We have $height = \alpha D$ and $width = \sqrt{1-\alpha^2}D$ where D is the diameter of the disk and α is a variable, $0 < \alpha < D$.

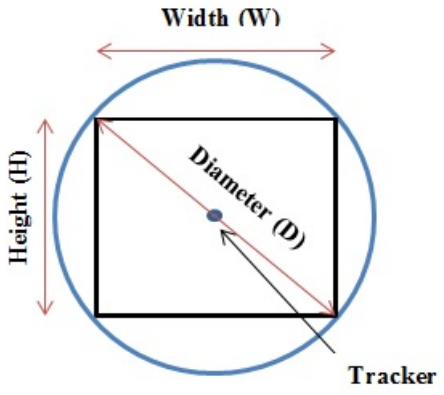

Figure 3.1: Rectangle Encapsulated Cover in Disk Model for Tracker

The diameter of the disk is given by $D = 2r$. Thus the tracker is able to sense the target only if it falls within the rectangle encapsulated in the disk's sensing diameter D. The sensors can be camera, sonar, laser, infra-red or any other sensors (depending on cost and environment) as long as it is able to detect the events within its radius.

We will set the base location of the trackers such that (i) In each of the base station there are as many trackers as there are targets and (ii) All the points in the observation area are within flying distance of trackers from atleast one base station.

If L is the set of base locations and A is the set of number of targets, then for our initial computation of our algorithm, the number of trackers needed would be

$|L| \times |A|$. During the real deployment, the number of trackers will be upper bounded by $|A|$, though we will try to minimize the number of trackers.

We consider the case where the target and trackers are both mobile with finite velocity, the trajectories of the targets is known, and the communication algorithm of trackers with base station for dispatch and routing is given. There is a long range control channel available to send information to trackers.

The trackers can identify their own location using GPS. Also for the sake of simplicity, the trackers have no constraints on energy (fuel, battery cells). With the increasing use of energy efficient drones and advances in high mileage endurance vehicles, it is a valid assumption.

The environment is generalized to stable environment with no wind. For similar weather conditions like heat, cold, rain; it is entirely dependent on the tracker's resilient structure and is out of scope for this research.

3.3 Problem Statement

Continuous tracking problem for mobile targets (CTPFMT): We are given a set (A) of 'n' targets $\{a_1, a_2, ..., a_n\}$ moving in a 2-D plane with known trajectories and velocities, and set (B) of trackers $\{b_1, b_2, ..., b_m\}$ with specified location bases $\{l_1, l_2, ..., l_i\}$, max velocity V_{max} and lead time t_L. Also, the base location of the trackers are placed such that (i) In each of the base station there are as many trackers as there are targets and (ii) All the points in the observation area are within flying distance of trackers from atleast one base station. The problem is to find the minimum set of trackers and their trajectories that satisfy the following requirements-

- Coverage - for every target a_i at any time $t \in \Re$, there exists at least one tracker b_i such that distance between them $dist(i, j) < r$, where r is the radius of the tracker.

- Mobility - for any tracker b_j with max velocity V_{max}, $\mid p(b_j, t) \sim p(b_j, t+1) \leq v \mid$

3.4 Geometric Disc Cover Problems

There is a similar problem which has been studied extensively in the past known as the Geometric Disc Cover problem (Gonzalez (1991)) in the context of static points. Given a set of points in the plane P, the problem is to identify a minimum cardinality set of fixed sized of disks (of prescribed radius) covering all points in P. Each point must be inside or on the boundary of atleast one of the disks in the cover. This problem can be generalized to d-dimensional space where the points will be in d-dimensions and the covering is by orthogonal hyperdisks of size D. These problems have many applications such as locating the least number of emergency facilities such that all potential users are located within a reasonable small distance from one of the facilities.

There are variations of this problem when the disks are replaced by squares or rectangles which has important application to image processing. An example is to store information in square patches, such that all pixels(points with information) are contained in at least one of the patches (Hochbaum and Maass (1985)).

All the problems described above are difficult to solve. They are strongly NP-Complete (Fowler *et al.* (1981)). Heuristics have been presented in (Tanimoto (1979); Tanimoto and Fowler (1980)) and polynomial time approximation algorithms are given in (Franceschetti *et al.* (2001); Srinivas *et al.* (2009); Hochbaum and Maass (1985))

This is similar to our problem when the $time \leq t_1$, where the targets(points) haven't started moving yet (they will move when $time > t_1$). The minimum trackers need to be identified and placed such that all the targets are covered at time instance t_1.

The authors in (Srinivas *et al.* (2009)) gave a 6 approximation algorithm (SCR) for static covering using rectangles, and 4 approximation algorithm (SCD) for static covering using disks. We will use the SCR algorithm in our proposed solution to our problem for static case(that is when $time \leq t_1$).

In the next section, we will reduce our problem to Geometric disc cover problem and prove that our problem is also NP-Complete.

3.5 NP Completeness Proof of our Problem

We will prove that our problem is NP-complete.

Theorem 1 CTPFMT is NP-complete

Proof: Consider the instance of GDC having set of targets on a geometric plane and fixed size objects(like disc, rectangles or squares).

Consider the instance of CTPFMT having set of targets (along with their trajectories and velocities) on a plane with available trackers (having maximum velocity and base stations).

We will show that GDC problem is a special case of CTPFMT problem and use the *restriction* technique to prove NP-completeness.

In CTPFMT problem we have a continuous time function (t) where $t_0 < t_1 \leq t \leq t_n$ and $t \in \Re$. Let $t_n = t_1$. Then we have only one instance of t which is t_1, which is a snapshot of continuous time. The positions of targets are static during that instance.

The maximum velocity of trackers is V_{max}. The trackers have to reach the locations (given by GDC solution) to cover the targets at time instance t_1. This V_{max} is given by $V = \frac{dist(i,j)}{t_L}$, where $dist(i,j)$ is the distance between base location of the tracker i and desired location of tracker j (from the GDC solution), and t_L is the Lead Time $= |t_1 - t_0|$ (difference between tracker movement time and target movement time). Let velocity be infinite, $V_{max} = \infty$ so that tracker can appear at the desired GDC

location instantaneously.

From these values, we have obtained a GDC problem from CTPFMT problem. Thus relaxing those values will make our problem a generalized version of GDC. Since we know GDC is NP-complete and is also a part of CTPFMT, we have showed that CTPFMT is also NP-complete.

Chapter 4

ALGORITHM FOR STATIC CASE

We solve the problem separately first allocating the trackers to the targets at the start of the observation time (when the targets are static) and then maintain the coverage till the end of observation time (when targets are mobile).

We also know that maximum number of trackers required is n. It may be optimal to place one tracker for each target. Thus we set the upper bound to the total number of targets required as the total number of targets.

We use the method that was introduced by Hochbaum and Maass (Hochbaum and Maass (1985)). The method used deals with the problem by (i) splitting the plane into equal size strips, (ii) solving the problem for each strip locally, and (iii) combining the local solutions to get a global solution. We apply this method to the observation region for our problem, where the trackers move around and have to be covered. See figure 4.1.

We also use the Static Cover with Rectangles(SCR) algorithm presented by (Srinivas *et al.* (2009)) to cover static targets. This algorithm provides an approximation ratio of 6 for covering of static targets using fixed size rectangles. We will use their solution which gives us the desired tracker location. We have to send the trackers from their base location to these desired locations and ensure feasibility of the solution. The algorithm is described below.

4.1 SCR Algorithm

The Strip Cover with Rectangles(SCR) Algorithm (please see 1) takes the input as a set of points (Targets) $A = \{1, 2, \ldots, n\}$ and their location in $x - y$ coordinates

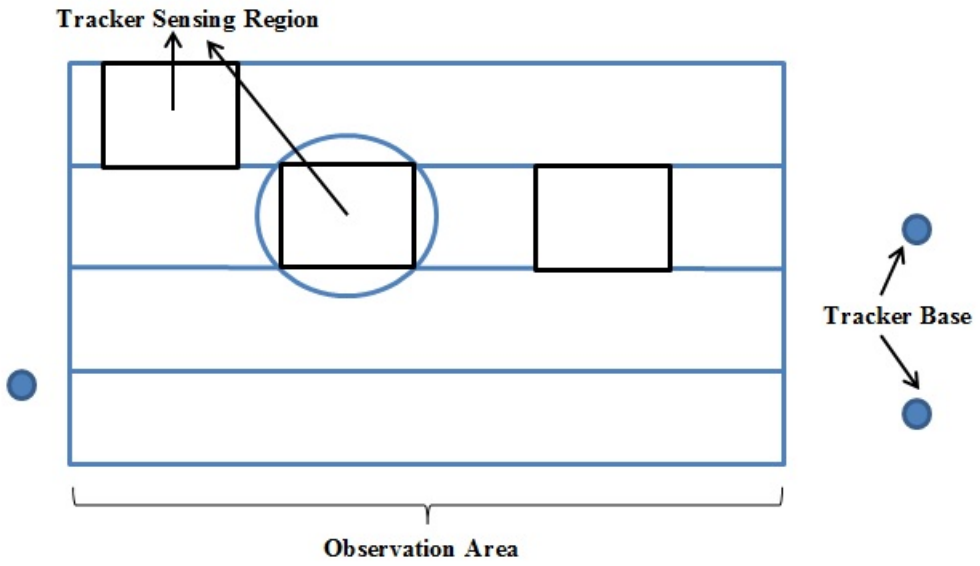

Figure 4.1: An Example of a Region with Target Observation Area and Tracker Base Stations

$(a_x, a_y) \forall a_x, a_y \in A$. The output is a set of disks (trackers) $B = \{b_1, b_2, \ldots, b_m\}$ and their locations such that all the points are covered.

The first step of the algorithm divides the plane in K strips of equal height $H = \alpha D$. The values of height that guarantee approximation ratio of 6 (derived in their paper) are $\frac{D}{2} \leq H \leq \frac{\sqrt{3}}{2} D$. Each strip is denoted by S_j and the trackers in strip S_j is denoted by B_{S_j}.

The algorithm will go through each strip and allocate targets to trackers in greedy fashion. The position of any tracker will be $x = a_L + \frac{W}{2}$ and y coordinate is middle of the corresponding strip, where a_L is the left-most target covered by the respective tracker. The computation complexity of this algorithm is $O(nlogn)$, because of the sorting of points by x-coordinate.

Algorithm 1 Strip Cover with Rectangles (SCR)

let $W = \sqrt{1-\alpha^2}D$ be width of the rectangle encapsulated in the disk (tracker)

divide the plane into S strips of height $H = \alpha D$

$B_{S_j} \leftarrow \phi, \forall j = 1, \ldots, S$

for all strips $S_j, j = 1, \ldots, S$ **do**

    while there exist uncovered target in S_j **do**

        let i be the leftmost uncovered target in S_j

        place a tracker b_k such that it covers all the targets in the rectangular area with x-coordinates $[i_x, i_x + W]$

        $B_{S_j} \leftarrow B_{S_j} \cup b_k$

return $\bigcup_j B_{S_j}$

4.2 Algorithm for Static Targets

In this section we propose an algorithm to solve the problem for the static targets (i.e. before the targets start moving), when the trackers have finite maximum velocity. We will try to find the minimum number of trackers (discs with prescribed radius r) which can cover a given set of b targets in a plane. Mathematically, given the set of points (A) distributed in a 2-D plane, place the smallest set of disks (B) such that for every point $i \in A$, there exists atleast one disk $j \in B$, such that $dist(i, j) \leq r$.

4.2.1 Description

Let us consider that the trackers are available at time t_0 and targets start moving at time t_1 where $t_1 > t_0$. There is a lead time $|t_1 - t_0| \neq 0$ available during which the trackers can take-off from their base locations and position itself to cover the targets before it starts moving. In order to compute the solution to this problem, we first model the input as a bipartite graph, and then solve bipartite graph maximum

matching to find a feasible solution and update the trajectory of the needed trackers required when the time is t_1.

We first give the definition of maximum matching for a bipartite graph (Even (2011)).

4.2.2 Maximum Matching in a Bipartite Graph

Given a undirected graph $G = \langle V = (L \cup R), E \rangle$, where there is no odd cycle, i.e G is bipartite and number of vertices are n, ($|V| = n = |V|$) and the number of edges are m($|E| = m$). We can divide V into two partitions, Left and Right (L and R), such that $\forall (u, v) \in E$, where $u \in L$ and $v \in R$. A matching M in G, is a set of pairwise non-adjacent edges, i.e. no two edges share a common vertex. A Maximum Bipartite Matching problem (also called as maximum cardinality matching problem) is a matching with the largest possible number of edges (that is also globally optimum).

We can find maximum matching M_{max} in G, by reducing it to a maximum network flow problem and solve using the Ford-Fulkerson algorithm (Ford and Fulkerson (1962)) in $O(VE)$ time. We will now define the maximum network flow problem.

4.2.3 Network Max Flow Problem

Given a capacitated network $G = \langle V, E \rangle$with a non-negative capacity $c(i, j)$ associated with each edge $e = (i, j)$, and two special nodes source s and sink t, ($s \neq t$). A flow is defined to be a function $f : E \to \mathbb{R}^+$ satisfying the following conditions:

$$\sum_{j \in V} f(i,j) - \sum_{j \in V} f(j,i) = \begin{cases} F, & i = s \\ 0, & i \neq s,t \\ -F, & i = t \end{cases}$$

$$0 \leq f(i,j) \leq c(i,j)$$

For some $F \geq 0$, where F is the value of the flow f. The network max flow problem is to maximize the total amount of flow from s to t subject to above conditions.

The solution to this problem can be found using Ford-Fulkerson Algorithm.

4.2.4 Ford Fulkerson Algorithm

The Ford-Fulkerson algorithm is used for solving maximum flow problem. Please see algorithm 2. This algorithm is implemented by using a *residual graph.*

Algorithm 2 Ford Fulkerson Algorithm

1: Let the initial flow f=0

2: **while** an augmenting path exists from source s to sink t **do**

3: $f = f + augmenting - path - flow$

4: return f

In the step 3 of the algorithm 2, max flow will be reached when no augmenting paths can be found in the graph. However, there is no certainty this case will be reached if the flow has irrational value. If the flow is integral then runtime of Ford-Fulkerson is $O(Ef)$ where E is the total number of edges and f is the maximum flow of the graph. Edmonds-Karp (Edmonds and Karp (1972)) is a variation of Ford-Fulkerson which guarantees termination and runtime of $O(VE^2)$ (as it doesn't depend on flow values), where V is the total number of vertices.

For our problem we will consider integral flow and use Ford-Fulkerson algorithm.

4.3 Graph Construction

We construct a bipartite graph $G = \langle V = (L \cup R), E \rangle$, where L and R are two partitions of V. We have set of idle trackers B available at time t_0 at predefined positions $p(b_i, t_0)$, ready for work.

Let L be the set of vertices that represent the idle trackers at their base stations at time t_0. The idle trackers are active trackers that are ready for work. Namely $L = \{v_0, v_1, \ldots, v_l\}$.

Let R be the set of vertices that represent the positions where the tracker is needed at time t_1 for covering the targets. These positions are found by running the SCR Algorithm over the given set of targets. Namely, $R = \{v_{l+1}, v_{l+2}, \ldots, v_n\}$. Each vertex $v_i \in B$.

There is an edge from a vertex in L to a vertex in R if and only if the tracker corresponding to the vertex in L can reach the position corresponding to vertex in R (also called as the feasibility to move to a new location). Let $dist(u, v)$ be the distance between tracker (position) u and tracker (position) v from L and R respectively. Then $E = \{(u, v) | dist(u, v) < u_{vel} \times \triangle t\}$ is a set of all edges in G; where $\triangle t = (t_1 - t_0)$ and u_{vel} is the maximum velocity of tracker at vertex u in L. Please see Figure **??** for an example. Note that, number of vertices in L may or may not be equal to R.

To convert our bipartite graph to network flow problem, we add new vertices s and d to V, such that there is an edge from s to every vertex in L and from d to every vertex in R. We keep all the capacities c on edges as 1. Then we solve the network flow problem on this graph. The maximum flow of this graph will correspond to the largest possible matching in G.

Note that the solution of maximum edges is only acceptable if it is equal to number of vertices in R. That is $|M_{max}| = |R|$.

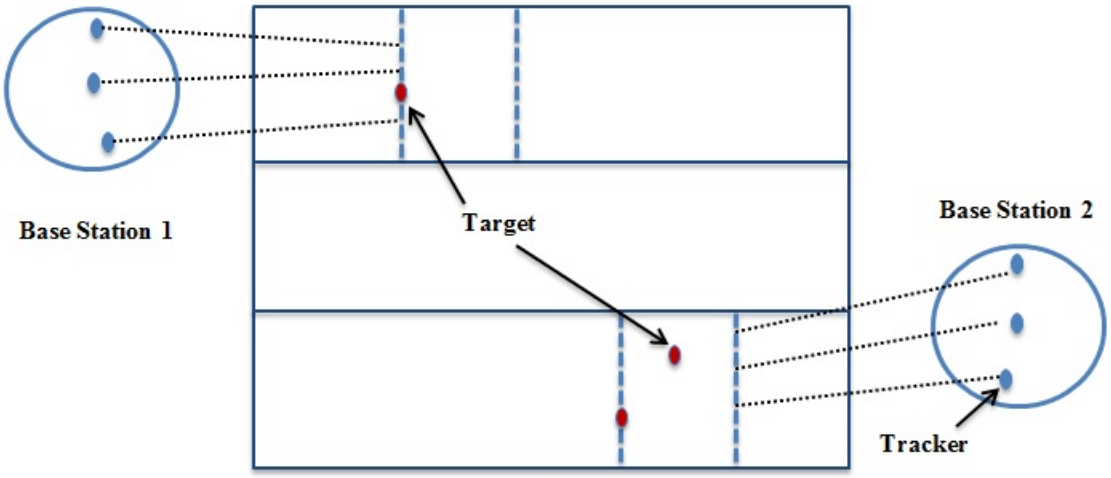

Figure 4.2: Covering of Static Targets at Time t_1

The solution shows that each position in R is occupied by a tracker in L which is incident on the corresponding edge, that is the tracker was brought from its base station to a new position to cover the targets. This new position is added as a waypoint to the respective tracker's trajectory. The cardinality of the solution gives the minimum number of trackers required. Please see Figure **??**

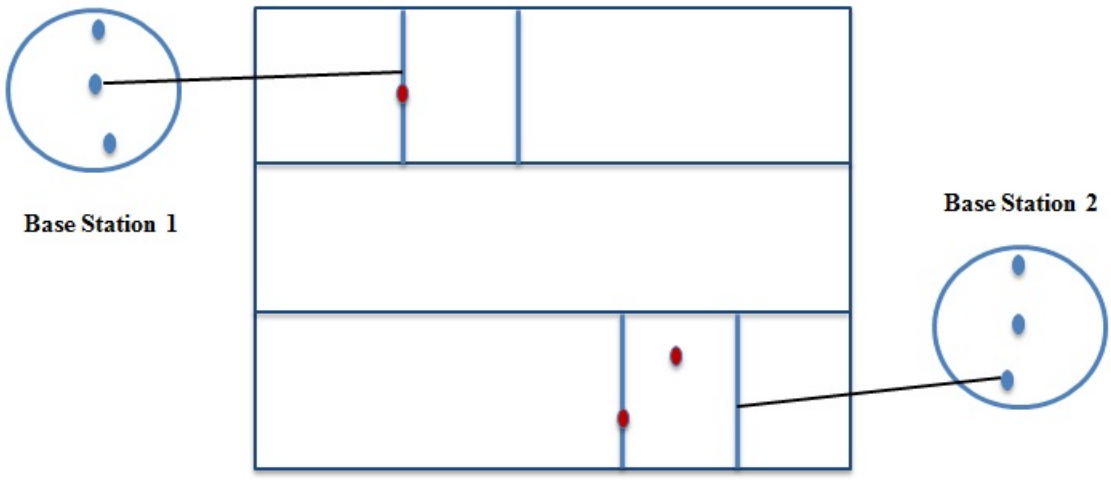

Figure 4.3: Trackers Selected for Covering of Targets at Time t_1

4.4 Algorithm PseudoCode

Please see Algorithm 3. In line 1, we keep track of three lists. The *activlist* maintains the trackers which are currently covering atleast one target. The *templist* keeps the imaginary trackers which are trackers that are assigned but not yet mapped to a real tracker. The *inactivlist* maintains the trackers which are idle, meaning it is not covering any targets. Let P_{b_k} be the set of targets covered by tracker b_k. We first use the SCR algorithm to find the positions to place trackers for covering the targets in each strip. Assume that imaginary trackers are placed in those locations. In lines 14-19 of our algorithm we replace the imaginary trackers with the inactive feasible trackers, and add them to *activlist* and remove them from *inactivlist*.

4.5 Analysis

We now analyze the time complexity for our proposed algorithm. The time taken by SCR algorithm is $O(nlogn)$, where n is the number of targets. The max-matching of bipartite graph G takes at the most $O(VE)$. In bipartite-graph, the left set of vertices have as many available trackers as base station times number of targets, that is $|L| = nL$. The right set of vertices have the positions where trackers are needed, that is the minimum trackers required when targets are static, hence in worst case $|R| = n$. Thus total vertices in G are $|V| = n^2L$. If the graph G is complete, then total number of edges $|E| = n^2$. Thus total complexity for our algorithm is $O(VE) = O(n^4L)$.

Also, we show that our static algorithm will always find a set of trackers that covers all the targets at time t_1.

Observation 1 The proposed static algorithm will always find a solution

Let's say we have n targets to be covered. Suppose there is target a_i which was left

Algorithm 3 Covering of Targets at Time = t_1

$templist \leftarrow \phi, activlist \leftarrow \phi, inactivlist \leftarrow \phi$

for all trackers b_k in B **do**

    $P_{b_k} \leftarrow \phi$ ▷ P_{b_k} - targets in cover of tracker b_k

divide the plane into S strips of width h where $\frac{D}{2} \leq h \leq \frac{\sqrt{3}D}{2}$ ▷ to guarantee 6 approx ration

for all strips S_j at time = t_1 **do**

    while there exists uncovered target in S_j **do**

        let i be the leftmost uncovered target in S_j, located at (i_x, i_y) position

        place the imaginary tracker c_k such that it covers all the targets in the rectangular region with x coordinates $[i_x, i_x + \sqrt{1-h^2}D]$

        $P_{c_k} \leftarrow P_{c_k} \cup i$

        $templist \leftarrow templist \cup c_k$

for all trackers in B **do**

    $inactivlist \leftarrow inactivlist \cup b_k$

Graph $G \leftarrow$ CreateBipartiteGraph($inactivlist$, $templist$)

if Maximum-Matching(G) = $|templist|$ **then**

    for all $e = \{b_k, c_k\} \in G$ **do**

        add the location of c_k as the next waypoint for b_k

        $P_{b_k} = P_{c_k}$

        $activlist \leftarrow activlist \cup b_k$

        $inactivlist \leftarrow inactivlist \setminus \{b_k$

else

    return 0 (error)

22: **procedure** CREATEBIPARTITEGRAPH(*inactivlist*, *templist*)

23: create graph $G = \langle inactivlist \cup templist, E \rangle$ where $E = \{\phi\}$

24: **for** all b_k in activlist and c_k in templist **do**

25: $E = E \leftarrow \cup\{b_k, c_k\}$ if $(V_{b_k} \times |t_1 - t_0| \leq d(c_k, b_k))$

26: return G

uncovered for some reason. This can happen because of two cases, (i) there was no edge to the rectangle that covers this target in the bipartite-graph from any tracker, (ii) all the incident edges weren't selected in the maximum-matching algorithm. Case (i) cannot happen because we had assumed that all the points in the observation area are within flying distance of trackers from atleast one base station. Suppose case (ii) is true. Then all the trackers (equal to n, since all base stations have n trackers from the assumption) in the base locations that could reach that rectangle are occupied by other targets. This would mean than number of targets covered by atleast one base station is n. But there was a target left uncovered, which would imply that there are $n + 1$ targets in the observation area. This is a contradiction.

4.6 Alternate Approaches

There are alternate algorithms for finding max matching on bipartite graph such as Hopcroft-Karp (Hopcroft and Karp (1973)) with time complexity $O(\sqrt{V}E)$, which can be used instead of Ford-Fulkerson. Since practical inputs for this problem is small and the overall complexity of complete algorithm is higher, we do not consider these optimizations. More info on practical comparisons can be found in (Setubal *et al.* (1996)).

Also, it is possible to have multiple solution that has maximum matching in our graph. 4.4 illustrates such an example. This means that the same number of trackers

can be used to cover targets but with different trajectories. To save tracker's energy, we could choose the solution which has lowest overall travel cost in moving tracker to it's new destination, i.e. one with minimum $\sum dist(i,j), \forall b_i \in L, \forall b_j \in R$. Note this does not mean that the number of trackers would reduce by choosing the least cost solution. This can be solved by putting the distance cost on the edges (replacing unit cost) and solving for minimum cost maximum bipartite matching. There are strongly polynomial methods for solving min cost max bipartite matching (Gabow and Tarjan (1989); Goldberg and Tarjan (1989); Kuhn (1955)) in literature. We have added this task to our future work.

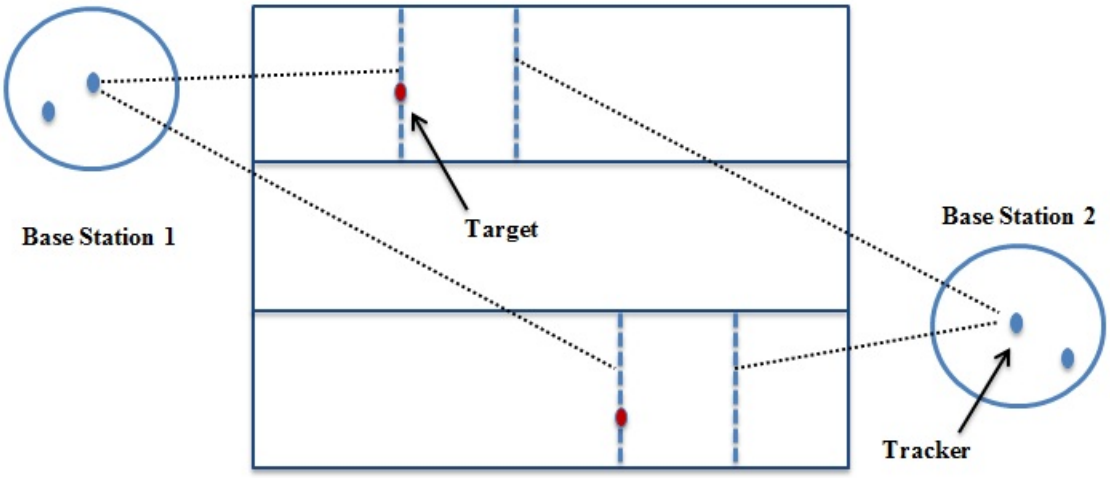

Figure 4.4: Multiple Solutions for Covering of Targets at Time t_1

Chapter 5

ALGORITHM FOR DYNAMIC CASE

5.1 Description

The algorithm 2 considers the case at the time t where $t_1 < t \leq t_f$. t_f is the end of observation time. After we have allocated the trackers at $t = t_1$, we handle the covering when the targets are set in motion. A target movement may change the allocation of targets to tracker along the whole strip. We develop an algorithm that are tailored to the frequent target movements.

Before describing the algorithm we recollect some definitions. We have P_{b_i} as targets covered by tracker b_i, for any $b_i \in B$. Also, a_i^L is the leftmost target covered by any tracker b_i, it's rectangle cover width $W = \sqrt{1 - H^2}D$ where H is the strip's height and D is the sensing diameter.

We maintain the following conditions throughout the cover- (i) Each tracker always has it's left rectangle cover boundary incident on the leftmost target it is covering, (ii) Tracker's cover domain is disjoint - $P_{b_i} \bigcap P_{b_j} = \phi \forall b_j \in B$, (iii) If any target a_i is within the boundaries of b_j, then $a_i \in P_{b_i}$.

If a tracker set gets narrow and comes close enough to another tracker's cover such that all its target can be covered in another's target rectangle, we free the tracker and add it to the *inactivList* along with the L set of the bipartite graph for reuse.

All the active trackers are assigned the same velocity and movement direction as the leftmost target in their cover. We say that an event has occurred when an target goes out of cover of its tracker (explained in detail in next section). All the events are added to Min-Heap and we handle them in the order of time of occurrence. We

will maintain a Min-Heap that stores these events to maintain partial ordering.

5.1.1 Min-Heap Data Structure

We will be using min-heap data structure for storing events(discussed in next section). A min-heap is a specialized tree based data structure. It is a binary tree such that (i) the value contained in each node is less that(or equal) to the value in it's children, and (ii) the binary tree is complete.

The heap is not necessarily sorted but is said to be partially ordered. The lowest element is always stored at the root of this data structure. It is very efficient for accessing min elements in constant time when the insertions and deletions happen at random. Among the various operations of min-heap, we will use find-minimum($\Theta(1)$), insert($\Theta(longN)$) and delete-minimum($\Theta(logN)$), where N is the number of elements in the data structure.

5.2 Events

Let P_{b_j} be the set of targets covered by tracker b_j and $a^L_{b_j}$ be it's left most target. An event $\xi(b_j, a_i, t_k)$ in our algorithm is the time instance t_k when a target a_i is the first target to go out of cover of it's tracker b_j. That is, the target a_i is currently lying on the boundary of the tracker's sensing region and will go out cover at next time instance, before any other target in P_{b_j} does.

The velocity and direction of the tracker is assigned the value of the leftmost target $a^L_{b_j}$'s speed and velocity. Thus, the time a target goes out is calculated by doing simple mathematics. The target is checked if it touches the border of the tracker on all the four sides (Top, Bottom, Left and Right). If $T_{out}(a_i)$ is the time when any target a_i gets on the border and is going out of cover of P_{b_j}, we will find $T^{min}_{b_j}$ given by

$$T_{b_j}^{min} = minimum(T_{out}(a_i)|\forall a_i \in P_{b_j})$$

Note that for the leftmost target $a_{b_j}^L$, because the tracker is moving at same velocity and direction, we only consider the events when it changes the strip (that is top and bottom border). The events are computed individually for each active tracker $(b_j|P_{b_j} \neq \phi)$ and are stored for later processing. We break ties based on the target ids.

Events in our algorithm are crucial to identify the sequence of changes happening during the motion of mobile targets. We store all the events in Min-Heap data structure (Cormen *et al.* (2001)) to maintain the time ordering for processing the targets going out of cover. After we have received events for all active trackers, we manage the event with the lowest time value first by removing the root from the min-heap. Since the other events have not occurred yet, we can safely handle the root event first and recalculate other events if needed. We categorize different events scenarios in two cases (see Figures 5.1 and 5.2). This covers all the cases when a target can go out of cover.

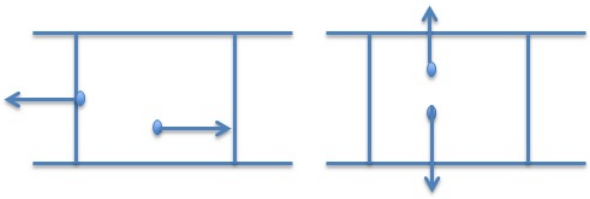

Figure 5.1: Case 1 for Event in Dynamic Cover

Case 1 events happen when a target in P_{b_j} is going to go out of cover on the right side of tracker b_j (the separation between its leftmost and this target becomes greater than cover length) or it is going to enter a different strip. We split of this target from b_j into a singleton set. We first check if there exists any tracker that can cover

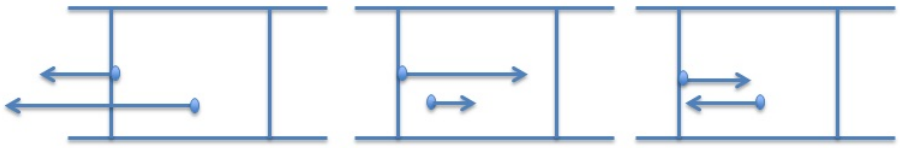

Figure 5.2: Case 2 for Event in Dynamic Cover

this singleton target. We assign the target to the existing tracker if it falls within it's cover, otherwise we will bring a new imaginary tracker at that location such that it's leftmost border is on this target and assign it's velocity and direction of motion to that of this target. We keep a track of time and position when this new imaginary tracker was added and put it in $|R|$ of bipartite-graph (similar to one we used before, so we can get a free tracker in it's place after doing maximum matching). Then we again check for new events in both new and old tracker and put them in min-heap.

Case 2 events happen when a target in P_{b_j} is going to go out of cover on the left side of tracker b_j thus changing the leftmost target of the tracker b_j. In this case, we update the leftmost target of the tracker and match the speed and direction of b_j with this target. We then calculate new event on this tracker and put it in min-heap.

We take care of overlaps that occur due to adding new tracker by using propagation technique described in next section.

5.3 Propagation

Whenever a new tracker is added to the strip to cover a future uncovered target, we must check for any overlaps with the existing trackers in the same strip (see Figure 5.3).

We resolve this by adding all the overlapped targets to the new tracker. The leftmost non overlapping target is made the left boundary of the conflicting tracker,

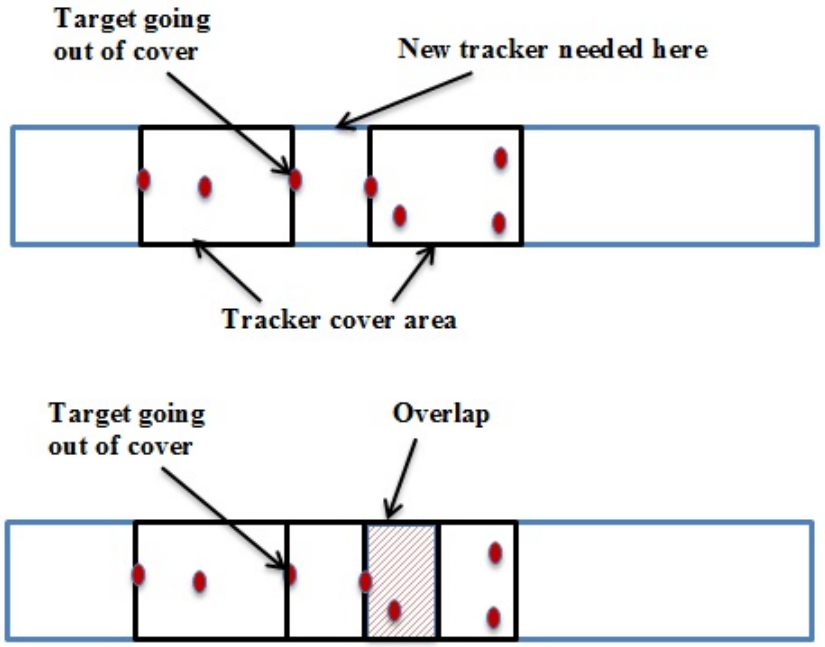

Figure 5.3: Propagation Example

and it is adjusted to move accordingly. If this movement causes overlap with another tracker, then the whole process is repeated again.

5.4 Algorithm PseudoCode

See Algorithm 4

Algorithm 4 Covering of Mobile Targets Where $t_1 < t \leq t_f$

Require: Set of all targets $a_i \in A$ and their waypoints.

ActivList and InactivList of trackers from Algorithm 1

Ensure: Minimum number of trackers and their waypoints

1: $MinHeap \leftarrow \phi, templist \leftarrow \phi, t = t_1$

2: **for** all trackers b in ActivList **do**

3: $vel(b) = vel(a_L^b)$ ▷ a_L^b is the leftmost target covered by tracker b

4: FindEvent(b)

5: **while** Time $t \leq t_f$, where $t \in \Re$ **do**

6: Event E = pop(MinHeap)

7: **if** E is case 1 **then**

8: **if** any adjacent tracker b can cover a_e, that is $(b_L)_x \leq (a_e)_x \leq (b_L)_x + w$ **then**

9: $P_b \leftarrow P_b \cup \{a_e\}$

10: **else**

11: place the imaginary tracker c_k such that it covers all the targets in the rectangular region with x coordinates $[(a_e)_x, (a_e)_x + w]$

12: $P_{c_k} \leftarrow P_{c_k} \cup a_e$

13: $templist \leftarrow templist \cup c_k$

14: Propagate(c_k)

15: **if** E is case 2 **then**

16: $a_e^L = a_e$ ▷ a_e^L is the leftmost target in cover of tracker b_e

17: $Vel(b_e) = Vel(a_e^L)$

18: $t = t + t_e$

19: FindEvent(b_e)

20: **if** $P_{b_e} = \phi$ **then**

21: $inactivlist \leftarrow inactivlist \cup b_e$

22: $activlist \leftarrow activlist \setminus \{b_e\}$

23: Graph $G \leftarrow$ CreateBipartiteGraph($inactivlist$, $templist$)

24: **if** Maximum-Matching(G) = $|templist|$ **then**

25: **for** all $e = \{b_k, c_k\} \in G$ **do**

26: add the location of c_k as the next waypoint for b_k

27: $P_{b_k} = P_{c_k}$

28: $activlist \leftarrow activlist \cup b_k$

29: **else**

30: return 0 (error)

31: **procedure** CreateBipartiteGraph($inactivlist$, $templist$)

32: create graph $G = \langle inactivlist \cup templist, E \rangle$ where $E = \{\phi\}$

33: **for** all b_k in activlist and c_k in templist **do**

34: $E = E \leftarrow \cup\{b_k, c_k\}$ if $(V_{c_k} \times |t_1 t_0| \leq d(c_k, b_k))$

35: return G

36: **procedure** FindEvent(b)

37: **for** every target $a_j \in P_b$ except a_j^L **do**

38: $T_{out}(a_j) =$ time a_j goes out of cover

39: $T_{min} = minimum(T_{out}(a_j) | \forall a_j \in P_b - \{a_j^L\}$

40: $MinHeap \leftarrow MinHeap \cup \xi(a_i, b, T_{min}$

41: **procedure** PROPAGATE(b)

42: **while** $b \cap b^R \neq \phi$ **do** ▷ b^R is tracker on immediate right of b

43: $P_b \leftarrow P_b \cap \{a_j | b_x \leq (a_j)_x \leq b_x + w\}$

44: $P_{b^R} \leftarrow P_{b^R} - \{P_b\}$

45: place the right tracker b^R such that it covers all the targets in the rectangular region with x coordinates $[(a^L)_x, (a^L)_x + w$

46: $b = b^R$

Chapter 6

VISUALIZATION

6.1 Visualization Software

Visual representations are important in human life. It allows us to perceive the spatial positions of elements and their relationship quickly. In the same reasoning with the proverb "A picture is worth a thousand words", humans grasp the content from pictures much faster than just text. Most of the recent studies in the covering and connectivity problems are void of any graphical visualization. It is useful to demonstrate the results obtained from the algorithm on a visual interface for conveying information to the target audience who are not well versed with the technical background.

Though the majority of the thesis is focused on developing the algorithm (where most of the intellectual work lie), the graphical user interface also contains some important contributions. In this section I describe the visual simulation of the target tracking problem.

6.1.1 Back-end

The back-end is the implementation of the algorithm defined in Chapter x. The entire code is written using Java.

The input to the program is the targets file "targets_waypoints.txt" along with configuration parameters. The tracker configuration parameters include the base locations, maximum speed (V_{max}), availability time(t_0), sensing diameter(D), and maximum available trackers $|B|$(which ideally should be equal to number of targets).

The target configuration parameters are speed(S) and number of targets to be tracked ($|A|$). Other parameters needed are boundaries of the planar region($0, 0$ to L, H), observation area ($X_{max}, X_{min}, Y_{max}, Y_{min}$) and total observation duration(t_n). The target file consist of the trajectory of each target. For each target(one number in entire line), there will be one or more waypoints on each consecutive line. The first waypoint is the starting position of the target. A waypoint is written as x and y coordinate with limits $X_{min} \leq x \leq X_{max}$and $Y_{min} \leq y \leq Y_{max}$. The targets are numbered from 0 to $|A| - 1$.

Sample 3 targets input file:

0

75 34

800 500

1

800 600

200 500

2

789 599

200 450

The output of the program is the trajectory of each tracker written in file "trackers_waypoints.txt". It is similar to the tracker input file with one change. Each waypoint is followed by velocity value for moving to that waypoint. The waypoint coordinates limits are $0 \leq x \leq L$ and $0 \leq y \leq H$. For the velocity of the tracker we have $0 \leq v \leq V_{max}$. The trackers are numbered from 0 to $|B| - 1$.

Sample 3 trackers output file:

0

36 4 70.0

811 588 70.0

1

1001 616 70.0

97 83 70.0

2

24 15 70.0

Note that we have used constant speed for each target for simplicity reasons. Variable speed can be easily added without the need to change the algorithm.

6.1.2 Front-end

Java swing, applet and awt libraries were used to implement the front-end for the visualization.

The input to the front end is the targets file "targets_waypoints.txt" and trackers file "trackers_waypoints.txt" .

The output is the java applet is a window which shows the live motion of the trackers and the targets. The targets are red and trackers are blue in color. All the targets covered by a tracker are shown by white lines from the target to the respective tracker. The boxes outside of observation area are the tracker base stations.

6.1.3 Interface

In this section, I present the screen-shot and give a brief description of how to use the software. See 6.1. On the side panel, you can change the simulation speed and refresh rate of the visuals. You can also view the number of trackers, targets, and radius of the tracker.

Figure 6.1: Software Interface Screenshot

Chapter 7

CONCLUDING REMARKS AND FUTURE WORK

In this thesis, I studied the problem of covering the mobile targets using mobile trackers in continuous space and time. We considered the case where trajectory of the target is known in advance and we came up with heuristic to find the minimum number of trackers required to track all the targets in static and dynamic cases.

In future, we would like to find the hardness and approximation bounds for our heuristic. Also, we would like to study the on-line version of the problem where trajectories of the targets are not known in advance. One of the variation of our problem could be to minimize the energy consumption of the trackers when tracking the targets. This can be done by replacing the maximum matching algorithm in bipartite graph with min cost maximum cardinality algorithm for finding the trajectory of the targets.

Also, another problem can be studied where if it is not possible to track all targets for the entire duration of observation with the trackers provided, then we try to minimize the periods of time when some targets were not tracked.

We made an assumption for our problem that atleast one tracker from any base location can reach any point in the observation area. If we remove this assumption then the problem becomes much harder. Let S_R be the region reachable by a target at the base station with given velocity and lead time. Let S_C be the region coverable by the same target depending on its sensing radius. We have $S_R < S_C$. With the rectangle cover, we can only cover points between S_L and S_R as shown in the figure 7.1 and miss the red target. However it is possible to cover the targets in the coverable but uncovered region by using disc model and adding more trackers as

shown in 7.2. In that case, the minimum trackers required might increase but satisfy our requirement. This scenario can be studied in the future work.

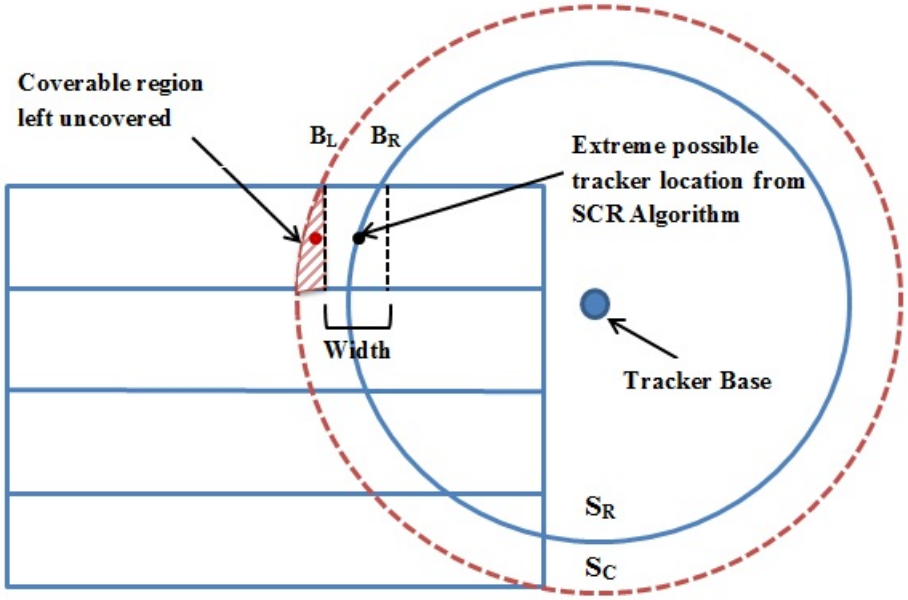

Figure 7.1: Covering by rectangles

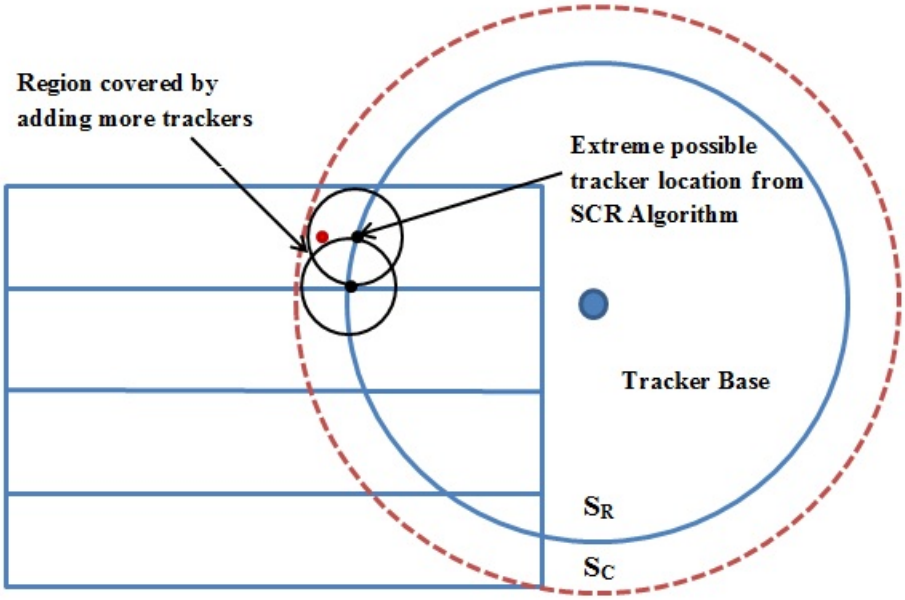

Figure 7.2: Covering by disc

REFERENCES

Adamey, E. and U. Ozguner, "A decentralized approach for multi-uav multitarget tracking and surveillance", in "SPIE Defense, Security, and Sensing", pp. 838915–838915 (International Society for Optics and Photonics, 2012).

Bai, F. and A. Helmy, "A survey of mobility models", Wireless Adhoc Networks. University of Southern California, USA **206** (2004).

Bar-Noy, A., I. Kessler and M. Sidi, "Mobile users: To update or not to update?", Wireless Networks **1**, 2, 175–185 (1995).

Bettstetter, C., H. Hartenstein and X. Pérez-Costa, "Stochastic properties of the random waypoint mobility model", Wireless Networks **10**, 5, 555–567 (2004).

Broch, J., D. A. Maltz, D. B. Johnson, Y.-C. Hu and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols", in "Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking", pp. 85–97 (ACM, 1998).

Camp, T., J. Boleng and V. Davies, "A survey of mobility models for ad hoc network research", Wireless communications and mobile computing **2**, 5, 483–502 (2002).

Chen, H., X.-m. Wang and Y. Li, "A survey of autonomous control for uav", in "Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on", vol. 2, pp. 267–271 (IEEE, 2009).

Chiang, C.-C. and M. Gerla, "On-demand multicast in mobile wireless networks", in "Network Protocols, 1998. Proceedings. Sixth International Conference on", pp. 262–270 (IEEE, 1998).

Cormen, T. H., C. E. Leiserson, R. L. Rivest, C. Stein *et al.*, *Introduction to algorithms*, vol. 2 (MIT press Cambridge, 2001).

Edmonds, J. and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", Journal of the ACM (JACM) **19**, 2, 248–264 (1972).

Even, S., *Graph algorithms* (Cambridge University Press, 2011).

Ford, L. and D. R. Fulkerson, *Flows in networks*, vol. 1962 (Princeton Princeton University Press, 1962).

Fowler, R. J., M. S. Paterson and S. L. Tanimoto, "Optimal packing and covering in the plane are np-complete", Information processing letters **12**, 3, 133–137 (1981).

Franceschetti, M., M. Cook and J. Bruck, "A geometric theorem for approximate disk covering algorithms", (2001).

Fujita, M. and A. Shimada, "Takeoff and landing control using force sensor by electrically-powered helicopters", IEEJ Transactions on Industry Applications **127**, 112–117 (2007).

Gabow, H. N. and R. E. Tarjan, "Faster scaling algorithms for network problems", SIAM Journal on Computing **18**, 5, 1013–1036 (1989).

Garcia-Luna-Aceves, J. and E. L. Madruga, "A multicast routing protocol for ad-hoc networks", in "INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE", vol. 2, pp. 784–792 (IEEE, 1999).

Garcia-Luna-Aceves, J. J. and M. Spohn, "Source-tree routing in wireless networks", in "Network Protocols, 1999.(ICNP'99) Proceedings. Seventh International Conference on", pp. 273–282 (IEEE, 1999).

Gloss, B., M. Scharf and D. Neubauer, "A more realistic random direction mobility model", TD (05) **52**, 13–14 (2005).

Goldberg, A. V. and R. E. Tarjan, "Finding minimum-cost circulations by canceling negative cycles", Journal of the ACM (JACM) **36**, 4, 873–886 (1989).

Gonzalez, T. F., "Covering a set of points in multidimensional space", Information processing letters **40**, 4, 181–188 (1991).

Hershberger, J., "Smooth kinetic maintenance of clusters", in "Proceedings of the Nineteenth Annual Symposium on Computational Geometry", SCG '03, pp. 48–57 (ACM, New York, NY, USA, 2003), URL `http://doi.acm.org/10.1145/777792.777800`.

Hochbaum, D. S. and W. Maass, "Approximation schemes for covering and packing problems in image processing and vlsi", Journal of the ACM (JACM) **32**, 1, 130–136 (1985).

Hong, X., M. Gerla, G. Pei and C.-C. Chiang, "A group mobility model for ad hoc wireless networks", in "Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems", pp. 53–60 (ACM, 1999).

Hopcroft, J. E. and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs", SIAM Journal on computing **2**, 4, 225–231 (1973).

Hsia, K.-H., S.-F. Lien and J.-P. Su, "Height estimation via stereo vision system for unmanned helicopter autonomous landing", in "Computer Communication Control and Automation (3CA), 2010 International Symposium on", vol. 2, pp. 257–260 (IEEE, 2010).

Hyytia, E., P. Lassila and J. Virtamo, "Spatial node distribution of the random waypoint mobility model with applications", Mobile Computing, IEEE Transactions on **5**, 6, 680–694 (2006).

Johansson, P., T. Larsson, N. Hedman, B. Mielczarek and M. Degermark, "Routing protocols for mobile ad-hoc networks-a comparative performance analysis", in "Proceedings of the 5th international conference on mobile computing and networking (ACM MOBICOM99)", pp. 195–206 (1999).

Johnson, D. B. and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks", in "Mobile computing", pp. 153–181 (Springer, 1996).

Kubota, Y. and Y. Iwatani, "Dependable takeoff and landing control of a small-scale helicopter with a wireless camera", in "Robotics and Biomimetics (ROBIO), 201 1 IEEE International Conference on", pp. 1279–1284 (IEEE, 2011).

Kuhn, H. W., "The hungarian method for the assignment problem", Naval research logistics quarterly **2**, 1-2, 83–97 (1955).

Le Boudec, J.-Y. and M. Vojnovic, "Perfect simulation and stationarity of a class of mobility models", in "INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE", vol. 4, pp. 2743–2754 (IEEE, 2005).

Naderan, M., M. Dehghan and H. Pedram, "Mobile object tracking techniques in wireless sensor networks", in "Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on", pp. 1–8 (IEEE, 2009).

Nitinawarat, S., G. K. Atia and V. V. Veeravalli, "Efficient target tracking using mobile sensors", in "Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2011 4th IEEE International Workshop on", pp. 405–408 (IEEE, 2011).

Radhakrishnan, G. S. and S. Saripalli, "Target tracking with communication constraints: An aerial perspective", in "Robotic and Sensors Environments (ROSE), 2010 IEEE International Workshop on", pp. 1–6 (IEEE, 2010).

Ravikiran, G. and S. Singh, "Influence of mobility models on the performance of routing protocols in ad-hoc wireless networks", in "Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th", vol. 4, pp. 2185–2189 (IEEE, 2004).

Rubin, I. and C. W. Choi, "Impact of the location area structure on the performance of signaling channels in wireless cellular networks", Communications Magazine, IEEE **35**, 2, 108–115 (1997).

Setubal, J. C. *et al.*, "Sequential and parallel experimental results with bipartite matching algorithms", University of Campinas, Tech. Rep. IC-96-09 (1996).

Srinivas, A., G. Zussman and E. Modiano, "Construction and maintenance of wireless mobile backbone networks", IEEE/ACM Transactions on Networking (TON) **17**, 1, 239–252 (2009).

Tanimoto, S. L., "Covering and indexing an image subset", in "Proc. 1979 iEEE Computer Society Conf. on Pattern Recognition and image Processing", pp. 239–245 (1979).

Tanimoto, S. L. and R. J. Fowler, "Covering image subsets with patches", in "Proceedings of the fifty-first International Conference on Pattern Recognition", pp. 835–839 (1980).

Wheeler, M., B. Schrick, W. Whitacre, M. Campbell, R. Rysdyk and R. Wise, "Cooperative tracking of moving targets by a team of autonomous uavs", in "25th Digital Avionics Systems Conference, 2006 IEEE/AIAA", pp. 1–9 (IEEE, 2006).

Xu, E., Z. Ding and S. Dasgupta, "Target tracking and mobile sensor navigation in wireless sensor networks", Mobile Computing, IEEE Transactions on **12**, 1, 177–186 (2013).

Yoon, J., M. Liu and B. Noble, "Random waypoint considered harmful", in "INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies", vol. 2, pp. 1312–1321 (2003a).

Yoon, J., M. Liu and B. Noble, "Sound mobility models", in "Proceedings of the 9th annual international conference on Mobile computing and networking", pp. 205–216 (ACM, 2003b).

Zhan, P., D. Casbeer and A. L. Swindlehurst, "A centralized control algorithm for target tracking with uavs", in "Conference Record of the 39th Asilomar Conference on Signals, Systems and Computers", pp. 1148–1152 (2005).

Zonoozi, M. M. and P. Dassanayake, "User mobility modeling and characterization of mobility patterns", Selected Areas in Communications, IEEE Journal on **15**, 7, 1239–1252 (1997).

Zorbas, D., T. Razafindralambo, D. P. P. Luigi and F. Guerriero, "Energy efficient mobile target tracking using flying drones", Procedia Computer Science **19**, 80–87 (2013).

Zou, Y. and K. Chakrabarty, "Distributed mobility management for target tracking in mobile sensor networks", Mobile Computing, IEEE Transactions on **6**, 8, 872–887 (2007).