Improving Surveillance using Cooperative Target Observation

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS
in Computer Science and Engineering
by Research

by

RASHI ASWANI
201102140
rashi.aswani@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled ”Improving Surveillance using Cooperative Target Observation” by Rashi Aswani, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Dr. Praveen Paruchuri

To My Family

Acknowledgments

I would like to sincerely thank my adviser Dr. Praveen Paruchuri, for his immense support, guidance and motivation. It would not have been possible to produce this work without his guidance. Working under his guidance has been a great learning experience. He has always shown enormous patience and belief in me, which helped me maintain confidence throughout the course of this work. I would also like to thank Prof. Kamalakar Karlapalem for his guidance and support during my early stages of work which shaped the work in the right direction.

I thank all my lab mates, especially Akash, Sravya, Navya, Yashaswi and Sai Krishna for encouraging me throughout my research career at IIIT and helping me with their valuable suggestions.

I thank my friends Swapna, Monam, Kriti, Chitra, Nikhar, Anirudh, Rohan and Hardhik for always being pillars of support, motivating me to work hard and making my stay a memorable one.

Finally, I thank my parents and siblings, Sona and Gaurav for their constant motivation, encouragement and having faith in me during my research.

Abstract

The Cooperative Target Observation (CTO) problem has been of great interest in the multi-agents and robotics literature due to the problem being at the core of a number of applications for example surveillance, inspection and search-and-rescue. In CTO problem, the observer agents attempt to maximize the collective time during which each moving target agent is being observed by at least one observer agent in the area of interest. The area of interest can include airport, border patrolling or public events. Micro-drones are typically used for such operations as with large gatherings and protests, or even when monitoring individuals, it is often difficult to maintain an overview. They can be pretty useful for surveillance allowing for a (top) view of the happenings on ground.

The CTO problem in most of the prior works is solved by finding the best suited strategy for observer agents, considering the target agent's movement to be Randomized. Given our focus on surveillance domain, we modify this assumption to make the targets strategic and present two target strategies namely Straight-line strategy and Controlled Randomization strategy. We then modify the observer strategy proposed in the literature based on the K-means algorithm to introduce five variants over the K-means algorithm and provide experimental validation across the strategies. In surveillance domain, it is often reasonable to assume that the observers may themselves be a subject of observation for a variety of purposes by unknown adversaries whose model may not be known. Randomizing the observer's action can help to make their target observation strategy less predictable, so that the unknown adversary cannot harm the observer agent. As the fifth variant, we therefore introduce Adjustable Randomization into the best performing observer strategy among the four variants over K-means algorithm, where the observer can adjust the expected loss in reward due to randomization depending on the situation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The word Surveillance means ”watching over”, sur means ”from above” and veiller means ”to watch”. Surveillance is the process of monitoring the area of interest. It can be done for the purpose of protecting people and property. Surveillance systems are becoming increasingly popular in the globalization process from static Closed Circuit Television systems (CCTVs), video recordings to micro-drones providing a 360 ° aerial view of the region. However, full involvement of human operators in traditional surveillance systems has led to shortcomings, such as high labor cost, inconsistency during long-duration.

1.1 Surveillance Systems

Security surveillance systems can be used in both large and small areas for border patrolling, airport security, rallies, home security and shopping marts, etc. They provide a complete overview of the activities taking place in the area of interest which helps track intruders in case of a security threat.

Surveillance systems have become very effective with their usage in control of crowds in dangerous areas. For instance, Israeli-built Heron Unmanned Aerial Vehicles (UAVs) [51] were deployed with the help of the Indian Air Force and Army for security of ”hot spot” locations in the Kashmir Valley in December 2014 elections. Drones were deployed over the voting booth area for security reasons till the voting procedure was complete. Drones were also used in October 2015 Bihar elections. For the August and October 2015 elections, mini-drones were used in Haiti. These instances present the use of micro-drones where human intervention can be quite dangerous.

Surveillance is used by government for the protection of people and property. Similarly, it is also used by criminal organizations to plan and commit criminal activities such as robbery and kidnapping, gathering information about the area of interest and others. For instance, in case of airport security, with the help of surveillance system, culprits can attain sensitive information of the area like number of security personnel deployed, their routines and use this information to plan their threat.

1.2 Cooperative Target Observation

Performing surveillance of suspicious people (targets) [8] [49] is an important task that security agencies across the world perform on a regular basis. The key difficulty here is that there are typically limited security resources for performing the Cooperative Target Observation (referred to as CTO in the literature). Some examples of security resources include static CCTV systems in closed areas, mobile ground robots or human resources in case of large open areas or micro drones for an aerial view if the area of interest is large enough to not be covered by mobile robots. While CCTV systems can be placed at multiple locations, they might not be able to cover the whole area of interest and the image feed will not be of high quality if the target is very far. There is another issue of sensor placement to determine where the sensors should be placed to maintain the view of targets [10] [52]. It is therefore important that the security resources are used in an efficient fashion.

The difficulty is exacerbated since the targets under surveillance themselves may have a variety of behaviors for example, people shopping in a shopping mart. [39] discusses the CCTVs installed in shopping malls used to monitor the behavior of people during shopping such as the shopping area, their head orientation and walking speed. These behaviors are used to find out the motivation of the user.

[24] presents an application where micro-drones are used to perform surveillance. In particular, with large gatherings and protests (or even when monitoring a set of individuals or wildlife), it is possible that there would be a number of people (or events) who could possibly be considered suspicious (called targets). Here, the aim of micro-drone would be to perform surveillance of the maximum number of targets (i.e. maintain an overview) instead of doing deep probe of a target i.e. spend all the time on a specific target who may not turn out to be problematic or dangerous.

[24] also discusses about risk free surveillance where the micro-drones use camouflage colors to reduce the risk of being sighted. This camouflage color helps in the safety of the drones. The motors used in these drones provide silent surveillance, unlike planes and helicopters which have a higher risk of being tracked down due to the sound.

With our focus on mass surveillance domain such as large gathering, we have two types of agents in our system namely Observer and Target. Observers are the agents performing surveillance of the environment. Targets are the suspicious entities which might be a threat to people or property e.g., terrorist or thieves in the real world. Observers do not have the prior information of the starting location of the targets.

We attempt to keep a track of all the targets for maximum duration of time. Observing the target activity prevents them from causing damage to the area of interest. We do not catch the target, rather we keep a track on its activity at all times. As we discussed above the limitations of static CCTV systems and video recording, we assume observers are mobile agents. For example, observers can be micro-drones having a 360 $^{\circ}$ aerial view of the field till their sensor range. Sensor range can be variable depending on the altitude of the micro-drone's placement. A target is said to be in observation when it falls inside the sensor range of an observer.

We propose five algorithms for observers namely, Explore-Exploit, Memorization, One-step Prediction, k-step Prediction and Explore-Exploit with Adjustable Randomization and two algorithms for targets improving over the Randomized Movement namely, Straight-line Movement and Controlled Randomization. Agents use these algorithms to calculate their destinations for further steps. At every step, a new destination is calculated for observer and target. We use K-means clustering algorithm or its variants along with observer's current position and past positions of targets which were present in observer's observation range (Memorization technique) to calculate the observer's destination for next step. Observers calculate their destination independent of other observers, i.e., there is no communication involved among them. Target destination is calculated based on the assumption that the target would like to maximize its distance from the observer observing it. The maximum distance is attained by following a straight line path away from the observer.

MASON simulation software is used to simulate the movement of the agents. Average number of targets present in the observer's observation range is the base criteria to compare the performance of the strategies presented. In this thesis, we develop a strategic component to obfuscate the aim of micro-drone wherein the surveillance strategy is inherently randomized with the amount of randomization being adjustable according to the situation.

1.3 Thesis contribution

Following are the contributions of this thesis:

- The thesis presents five strategies based on K-means and its variation for observers namely, Explore-Exploit, Memorization, One-step Prediction, k-step Prediction and Explore-Exploit with Adjustable Randomization to maximize their observation of targets and two strategies for targets namely, Straight-line Movement and Controlled Randomization strategy to make targets intelligent, taking into account the limited knowledge of the environment they have.

- Developed a simulation based on the MASON simulation software depicting the two agents behavior for the proposed strategies.

- We present a detailed experimental analysis to demonstrate effects of parameters like sensor range, speed of target agents and density of both the agents on mean number of targets observed by the observer.

- We introduce entropy (randomness) into the system in order to reduce the predictability of the observer's action as an unknown adversary can cause harm to the observer agent (drones) once it is able to predict its behavior.

1.4 Organization of the thesis

The rest of the thesis is organized as follows:

- **Related Work:** Chapter 2 presents background work on Cooperative Target Observation. It describes and analyzes various strategies developed for target observation over time. It also discusses the application of target observation and usage of micro-drones for surveillance. Art gallery problem and Pursuit Evasion problem plus its variants are also presented.
- **Model Description:** Chapter 3 presents the five strategies based on K-means and its variation for observer namely, Explore-Exploit, Memorization, One-step Prediction, k-step Prediction and Explore-Exploit with Adjustable Randomization and two strategies for target namely, Straight-line Movement and Controlled Randomization.
- **Experiments:** Chapter 4 describes the model, the parameters used for experimentation namely, density, speed of target agents, sensor range of observer agents and presents the detailed set of experiments performed along with comparison of the various strategies.
- **Conclusion:** Chapter 5 concludes the thesis.

Chapter 2

Related Work

Our work deals with the subject of improving surveillance using cooperative target observation. We identified the following threads of work related to it:

2.1 Applications of Target Observation

The Cooperative Target Observation problem has been of great interest to both multi-agents [15] [17] [38] and robotics [12] [43] [53] [29] literature due to the problem being at the core of many applications. [15], [38] present the major applications of Unmanned aerial vehicles (UAVs) for ground-target search in civil and military applications, such as environmental monitoring, battlefield surveillance, map building and others.

2.1.1 UAVs for Ground-target search

[15] presents an algorithm to search multiple static ground targets using UAVs. The authors divide the surveillance area into cells and each agent maintains its own probability map for the presence of targets in its cell. Agent updates their probability map individually using the Bayesian rule. The maps are then simplified to a linear update. They proved that all the individual cell's probability maps converge to one which shows the presence or absence of target in that cell. They also present a path planning algorithm for coverage optimization and connectivity maintenance.

[38] presents an approach for cooperative search of ground targets by distributed agents or UAVs. The agents communicate to each other, but have limited sensor range, fuel and time constraints. They attempt to search the dynamic targets in order to evade threats and coordinate strike against them. The algorithm has two inter-dependent tasks: (i) Learn the environment and store this information in the form of a search map. (ii) Compute an on-line guidance trajectory for the agent based on the search map.

[43] presents an approach to observe a ground target from an UAV by taking into consideration wind, camera limitations and UAV's performance. They provide a path geometry and camera angles for the

UAV to get the target in a constant line-of-sight relative to the UAV. They also develop a guidance law based on "good helmsman" behavior using minimal heuristics.

[12] presents an approach for the robot agent to reach a desired position or explore an area in an unknown environment. They present a system which has a combination of ground robot and an aerial robot coordinating among themselves to get an overview of the unknown territory. The aerial robot equipped with a camera helps the ground robot in navigation. The ground robot overcomes obstacles with the help of feedback received from aerial robot. The system was tested in an outdoor environment with a medium-sized ground robot and a mini quad-rotor and also in a simulated environment.

2.1.2 Surveillance

Applications also include mobile robots to survey and patrol large unstructured and unknown environments. This task is one of the objectives of the ROTOS project "RObot Team for Outdoor Surveillance" [42]. In the area of multi-robot surveillance, Naval Command Control and Ocean Surveillance Center have developed a security control system, MRHA (Multiple Robot Host Architecture) [11] for surveillance of warehouses. The system provides a single host console to control multiple vehicles. It is distributed and it can accommodate upto 32 robots. It is a semi-autonomous system, that uses human intervention when needed.

2.2 Microdrones

Micro-drones equipped with a camera play a vital role in providing the area information in Search and Rescue operations of an unknown terrain. The aerial view provided by the drones helps the recovery team to track targets (people) avoiding the dangerous areas.

[40] and [41] present the ongoing research on deploying camera equipped UAVs for disaster management applications. The UAVs are connected wirelessly and they work together to achieve a mission. Their main aim is to get a full view in the form of images or videos of the disaster prone area such as wood fire and traffic accident. These images and videos are then analyzed to get a complete picture of the disaster.

2.3 Security Games

Security agencies have an important task of protecting critical infrastructure across the world [48] [34]. As presented in [4], Bayesian Stackelberg games have played a key role in deployed security applications at a number of locations: ARMOR has been deployed at the Los Angeles International Airport (LAX) to randomize checkpoints on the roadways leading to the airport and canine patrol routes within the airport terminals since 2007 [35] [37], US Federal Air Marshal Service (FAMS) deployment has been randomized using a game-theoretic scheduler, IRIS [50] while PROTECT system was deployed

for generating randomized patrol schedules for the US Coast Guard in Boston, New York, Los Angeles and other ports around the US [3] [46]. Many other applications have been described in [4]. All these works use Bayesian Stackelberg games at the core, which compute a (mixed) strategy for the agent (leader here) which is fixed.

The CTO formulation on the other hand is inherently different in the sense that the strategy for the agent is generated dynamically every few time steps taking into account its current observation (of the position and other characteristics of targets). Furthermore, we do not require a detailed payoff model of the opponent for all the circumstances (for that matter even for the agent), instead work with local behaviors e.g., what may be the possible actions the opponent would have taken in the last couple of steps and other information requirements needed by each of the strategies described in the thesis.

2.4 Art gallery problem

The target observation work is most closely related to the cooperative multi-robot observation of multiple moving targets (CMOMMT) problem [31] which is a broader version of art gallery and related problems [27]. The basic art gallery problem is to determine the minimum number of guards required to ensure the visibility of an interior polygonal area. There are a number of the art gallery problem, involving static or mobile guards who ensure that the interior polygonal area is always visible. [13] presents a randomized algorithm that solves a variant of the art gallery problem [45] [52].

2.5 Cooperative Multi-Robot Observation of Multiple Moving Targets

[30] introduces the Cooperative Multi-Robot Observation of Multiple Moving Targets (CMOMMT) formalism which is similar to CTO problem. The CMOMMT problem is defined as follows. Given:

S : a two-dimensional bounded region

V : a team of m mobile robots, v_i, i = 1, 2,..., m, having a 360 ° view of the environment for a limited sensor range.

$O(t)$: a set of n targets, $o_j(t)$, j = 1,2,...,n.

A robot v_i, is observing a target when the target is within the sensor range of the robot v_i:

A mXn matrix $B(t)$ defining the target is under the observation the robot is as follows:

$B(t) = [b_{ij}(t)]_{mXn}$ such that $b_{ij}(t)$ =

$$\begin{cases} 1 \text{ if robot } v_i \text{ is observing target } o_j(t) \text{ at time } t \\ 0 \text{ otherwise} \end{cases}$$

The goal of the robot agents is to maximize the mean number of targets being observed by at least one robot agent throughout the mission of length T time units.

The research paper [30] presents an algorithm to minimize the total time by which targets escape from robot's observation. It takes into consideration the targets present in the observation range of the robot and the targets which are most likely to enter its observation range. The algorithm updates the local force vectors which are weighed according to the following conditions:

- Robots get attracted to the nearby targets.
- Robots get less attracted to the targets which are already being observed by some other robot.
- Robots repel each other so as to avoid collision among themselves.

In the A-CMOMMT algorithm discussed in [30], when a robot does not find any target nearby, the weighted sum of the force vectors causes the robot to move away from its nearby robots and then stay static in one location.

[47] addresses the problem of searching a target when there is no target near the robot by the development of search schedules for "∞-searchers". An "∞-searcher" is a mobile searcher having a 360 $^\circ$ view of the area of interest which can help the robot find a target by providing a search schedule. A search schedule is a path across the area of interest which allows the robot to detect a target. Another approach to this problem would be to focus the robot's search on areas which are previously known to have higher density of targets. The authors compared the A-CMOMMT approach to three other approaches: (i) Fixed, where the robots are static and are distributed uniformly over the area of interest, (ii) Random, where the robots follow a random/linear movement, (iii) Local, where the robots have non-weighted local force vectors. They tested their work with both simulation environment and physical robots.

ALLIANCE [28] demonstrates a fully distributed, behavior-based architecture for fault tolerant multi-robot cooperation. In this architecture, a team of robots have a mission to perform which is composed of sub-tasks, where the sub-tasks can be ordered as desired. Each team has a high-level function which is performed during the mission. The robots in the team can also select actions based on requirements in the middle of the mission such as, due to environmental conditions or other robot's behavior or some mechanical failure. This leads to adaptive action selection by the robots. In unpredictable and dangerous environments, this architecture helps robots to respond robustly, reliably, flexibly and coherently to change. The architecture was demonstrated through implementing a laboratory version of hazardous waste cleanup.

2.6 Pursuit Evasion

Pursuit Evasion [25] is a family of mathematical problems in which one group attempts to track the other group. It is a variant of cop and robbers game [6] [1].

A basic example of Pursuit Evasion in a graph is as follows: Pursuers and Evaders occupy nodes of a graph. They take alternate turns to either move along the edge to the adjacent node or stay at the same node. If a node previously occupied by evader is occupied by a pursuer, the evader is captured

and removed from the graph. The main difference between Cooperative Target Observation and Pursuit Evasion is that in Pursuit Evasion, the aim of the pursuer is to capture the evader and the evader is out of the game when caught, whereas in Target Observation the goal of the observer is to observe the target. If the target falls in the observation range of the observer, the observer keeps observing the target. There is no real end state for the game other than a time based deadline.

Pursuit Evasion games are used in robotics for studying motion planning problems, such as catching burglars or playing hide-and-seek. Pursuers and evaders have their own set of behaviors in such problems. They are also used to calculate the worst case scenario performance for a problem with appropriate behavior modeling.

In Robotics, Pursuit Evasion games can be solved with two approaches: differential and combinatorial [9].

The differential approach is used for solving non-cooperative differential games [5]. In these games, differential equations which calculate the movement of agents are brought together using Hamilton-Jacobi-Isaacs (HJI) differential equations. The solutions to HJI equations provide strategies for agents to achieve their targets. Advantage of this approach is that physical constraints like velocity can be modeled as differential constraints. This approach also has a disadvantage that the equations become rather complex to solve.

Combinatorial approach is generally used where the environment is very complex. A common approach for such games is to represent the complex environment geometrically and solve the game directly using this representation. Graphs can also be used to represent the environment topologically.

2.6.1 Pursuit Evasion on graphs: The cops and robbers game

In this game, cops are the pursuers who try to catch a robber (evader) by moving along the nodes of a graph. The agents move in turns on the nodes along the edges. If a node currently occupied by a robber is occupied by a cop, cops wins the game. This game was introduced by [26] and [1]. A simple dynamic programming algorithm can be used to solve such games which works as follows:

Let $G = (V, E)$ be the graph representing the game. For each vertex $v \in V$, $N(v)$ denotes its neighborhood $N(v) = u : (u, v) \in E, u, v \in V$. For a single cop, the algorithm starts by marking the state (u, u) as capture. For all unmarked states (c, r), if $\forall\ r' \in N(r)$, $\exists\ c' \in N(c)$ such that (c', r') is marked, then mark (c, r), until no further marking is possible. The game continues further like this. If the node occupied by the robber is marked, the cops win.

2.6.2 Parson's game

Parson's game [33] is a variation of Pursuit Evasion in a graph where the edges are not of equal length. The turn based approach does not work in this scenario as the Evader and Pursuer do not move equal distance. One approach considering the worst case scenario is to consider the evader to be very powerful and faster than the pursuer. This results in an "infinite speed model" where the edges represent

a tunnel in which an evader can be hidden. [32] defines a search number of the graph as the minimum number of pursuers necessary for such a capture.

2.6.3 Lion-and-man game

The Loin-and-man game is a variation of the cops and robbers game. In this game, the area of interest is a circular arena of radius r. The speed of the players is equal and is set to the maximum as one. In the cops and robbers game, the aim of cops is to move to the node occupied by the robber. Similarly aim of loin in this game is to capture the man by moving onto his location. One approach taken by Lion could be to start at the center of the circle and move on the radius which points to the man's position. [44] shows that the time taken with this approach is $O(r^2)$. Though [21] shows that with this approach the lion is not able to capture the man in continuous time frame. The man is able to escape if the game is played in continuous time. [2] presents a strategy with which the lion can get within a distance c of the man in time $O(r \log \frac{r}{c})$.

2.6.4 Pursuit Evasion in Robotics for graph

[20] presents a variation of Pursuit Evasion with complex indoor environment. The algorithm detects the intruders in the graph and is referred to as GRAPH-CLEAR. The GRAPH-CLEAR problem has a weighted graph with nodes comprising of arbitrary shaped areas and the edges act as transition between these nodes. The robot team can perform sweep and block actions on the nodes and edges. A sweep action is to detect the intruder present in a node region while a block action is to block the movement of intruders among the edges. A strategy is composed of sequences of sweep and block actions. The goal is to find the optimal strategy which uses the least number of robots. Nodes are labeled with the number of robots required to sweep the area and edges are labeled with the number of robots required to block the edge. [20] shows that the minimum strategies required for the graph can be found efficiently (polynomial-time) when the graph is a tree and is NP-hard otherwise.

2.6.5 Pursuit Evasion in polygonal environments

In a bounded continuous environment, the aim of the pursuer is to find a path which leads to the evader. The environment consists of obstacles and pursuer has a limited visibility of the environment. The evader's initial position or the current position is unknown to the pursuer. [16] presents randomized pursuer strategies for such a problem explaining how deterministic approach will not work in these scenarios.

[14] presents a greedy approach for a swarm of agents pursuing the evaders. The algorithm points pursuers to the direction having maximum probability of finding an evader at that particular instant. With this approach pursuers are able to find evaders in finite time under mild assumptions.

2.7 Cooperative Target Observation

Another domain very closely related to our work is [31] [23]. They study the Cooperative Target Observation (CTO) problem and compares the performance of K-means and Hill Climbing algorithms on centralized, partly-decentralized and fully decentralized agent strategies under different levels of target speeds, sensor ranges and update rates. The paper shows that K-means performs quite well over a large part of the parameter space tested and is pretty robust to the degree of decentralization.

The CTO problem as stated in [23], has a set of mobile agents called observers, which collectively attempt to observe as many targets as possible. Targets are assumed to move randomly in the environment. Observers have a limited sensor range and can observe targets which fall within a circle of radius R centered at each of the observer. The environment is a non-toroidal rectangular continuous 2D field free of obstacles. They present performance comparisons of centralized, partly-decentralized and fully decentralized CTO algorithms for a variety of parameter settings. The decentralized version retains the key characteristics of the centralized CTO problem with the main distinction that each observer takes its decision independently with the help of its local knowledge.

[19] presents an algorithm for the CMOMMT problem which utilizes information from sensors, communication and presents a mechanism to predict the minimum time before a robot loses a target. [18] proposes an algorithm that takes into account the densities of observers and targets. By manipulating these densities, a control law is proposed for each observer. In this thesis, we build upon the CTO problem formulation presented in [23] and adapt it for surveillance domain which needs us to focus on the following aspects:

- Model realistic behavior for targets
- Develop strategy for the observers to perform optimal target observation
- Randomize the observer's actions to make their target observation strategy less predictable.

We model the surveillance domain as a decentralized CTO problem in our work. Given our focus on surveillance a realistic model for target would be to assume that they are strategic. Furthermore, no assumption was made on the targets sensor range earlier since targets were assumed to move randomly and do not consider inputs from environment. We modify this assumption to make targets strategic with a sensor range similar to observers. Targets can therefore identify the presence of observer(s) within their sensor range. In particular, we model targets as intelligent agents, whose aim is to minimize their chances of being observed while observers aim is to maximize their target observation. Prior work on decentralized CTO shows that a K-means based solution [23] produces a high quality movement behavior for the observers. We build upon the K-means solution to introduce five variant strategies for the observer that consider the strategic behavior of the targets.

2.8 Summary

In this Chapter we discuss the significance of target observation in surveillance and search-and-rescue operations in civil and military applications where human inference might be dangerous. We present how micro-drones can be used efficiently in such scenarios. We study several approaches for Cooperative Target Observation and Pursuit Evasion. We also see how art gallery problem is closely related to our work. In our work, we have focused on developing strategies for the agents to improve surveillance. We also compare Pursuit Evasion with our work and found the main difference is, that the aim of the pursuer is to catch the evader and the game stops as soon as the evader is caught, whereas in our work, the observation still goes on even when the target is under observation. In Chapter 3, we discuss the strategies for observers and targets we developed. In Chapter 4, we present the experimentation comparing the various strategies and identify the best suited strategy for observer and target. We also study the effects of various parameters in our system.

Chapter 3

Model Description

3.1 Observers Movement

As described in [23], a K-means based algorithm can provide a high quality movement behavior for observer. The K-means based algorithm computes a destination point for each observer. The observer then moves towards this destination for γ time-steps or steps (with default value of 10). If the observer reaches its destination before γ steps, it waits until a new destination is computed. There is no communication involved among observers (or targets). Observers do not send any information but are synchronized in decision making due to the waiting time of γ steps (as modeled in [23]).

3.1.1 K-means based Algorithm For Decentralized CTO Problem

K-means is one of the simplest algorithm to perform clustering of unlabeled data points. Clustering is a process of dividing the unlabeled data points into clusters having similar data points. In K-means algorithm, the data points are divided based on distance. The data points close to one another form one cluster. It partitions N data points into K clusters where each data point belongs to the cluster having the nearest mean.

In our context, the target agents are the data points to be clustered and the observers are the cluster centers. The set of targets that fall within the range of an observer can be considered as a cluster. Note that clusters need not be of equal sizes. The goal of the algorithm would be to identify a new destination for each observer so that the cluster associated with an observer is more uniformly spread around it. This will improve the possibility of retaining the maximum number of targets over time.

Following are key steps of the K-means based algorithm as presented in [23]:

- Firstly, the algorithm obtains the number of clusters to pick which is same as the number of observers $|O|$ in our domain and then initializes the cluster centers with the observers initial positions.

- For each of the clusters $c1, c2, ..., c|O|$, it identifies the targets whose distance is less than the sensor range of the observer associated with that cluster. Observers observe all the targets that fall into their cluster.

- The algorithm then computes the mean of the targets present in each cluster ci and sets the destination position of the corresponding observer i closer to the mean using the following equation:

$$K_i = (1 - \alpha)K_i + \alpha M_i \tag{3.1}$$

  where K_i is the initial position of observer i, α is the weighing factor that takes into consideration the observer's initial position and M_i is the mean position of targets present in the cluster ci.

The algorithm iterates over the above mentioned steps till the time-steps (set as 1500 in our experiments) are exhausted. The destination for each observer is updated every γ steps. Therefore each update step for the observer consists of γ steps and is called update rate of the algorithm. At the start of new update step the algorithm sets a new (next) destination for each observer and allows the observers to act (i.e. move or wait) for γ steps that would help them reach their destination. Hence each observer keeps adjusting its current position that would enable it to retain more targets in its sensor range over time.

We now build upon the K-means based algorithm for observer to develop:

- **Explore-Exploit**
- **Memorization**
- **One Step Prediction**
- **k-step Prediction**
- **Explore-Exploit with Adjustable Randomization**

3.1.2 Explore-Exploit

In Explore-Exploit strategy, an observer has two specific actions namely Explore and Exploit. When there is no target present in observer's observation range, the observer performs an exploration of the environment in search of a target agent, this action is known as Explore action. In case there is a target present in observer's observation range, the observer performs the above described K-means based algorithm for Decentralized CTO Problem for determining its next destination which is known as Exploit action. It performs an Exploit action using the K-means clustering i.e. Eqn. 3.1 and can perform an Explore action by setting its destination position as a random point within a quarter of the environment width and height centered on the observer. The destination of observer is computed using the following formula:

$$\begin{aligned} &K_i = (1 - \alpha)K_i + \alpha(W_{explore} * RP_i + W_{exploit} * M_i) \\ &W_{explore} + W_{exploit} = 1 \end{aligned} \tag{3.2}$$

Here, K_i denotes the initial observer position, $W_{explore}$ denotes weight factor for Explore component, $W_{exploit}$ denotes weight factor for Exploit component, M_i denotes the mean position of targets present in i^{th} cluster and RP_i denotes a random point. Taking cue from prior work [23] on the method to pick a random destination for observer, we bound the randomly picked destination point RP_i for the observer to be within a quarter of the environment width and height centered on the observer. The intuition for picking dimensions for the rectangle in this fashion is to avoid observers getting clustered near the center of the environment while moving to their intended destinations. Note that the algorithm still follows the steps of K-means algorithm and hence is iterative i.e. the observer would take γ steps (either random or towards a destination) and then tries to identify a new destination point. Observer has autonomy to set values for the parameters $W_{explore}$ and $W_{exploit}$. We now present three settings for the Explore-Exploit strategy, generated by changing the weights for Explore and Exploit actions:

- **Model 1 (0-1 model)**: In this model, if the observer is not observing any target currently it performs an Explore action in search of targets i.e. $W_{explore}$ = 1. On encountering one or more target(s), it stops Explore and takes Exploit action of observing the target(s) using K-means clustering algorithm. Hence, if there is no target under observation, $W_{explore}$ = 1, otherwise $W_{explore}$ = 0.

- **Model 2 (0-.5-1 model)**: Model 2 sets $W_{explore}$ to either 0, 1 or 0.5. In case of no target under observation, $W_{explore}$ = 1, if more than two targets are being observed $W_{explore}$ = 0 and if either one or two targets are being observed, $W_{explore}$ = 0.5.

- **Model 3 ($(\frac{1}{|targets_i|+1})^2$ model)**: Similar to the above models, if the observer i is not observing any target it performs an Explore action. Upon encountering target(s), it assigns weight to the Explore component depending upon the number of targets being observed. In particular, the weight for Explore is set as $(\frac{1}{|targets_i|+1})^2$ i.e., as the number of targets being observed increases, the weight for Explore gets smaller and vice versa.

3.1.3 Memorization

In Memorization strategy, we try to improve the 0-1 Explore-Exploit strategy (reason for picking 0-1 Model is explained in the Experiments section of Chapter 3) by keeping track of the last closest target position that was in range and use this position to calculate observer's next destination when there is no target present in its observation range. The reason behind it is that observer hopes that the target may not have gone far from earlier and it may be able to capture it from its last position.

The algorithm goes in the following fashion: At the start of every update step ($us(i)$: update step i), each observer finds targets in its sensor range. If targets are present it performs an Exploit action of Explore-Exploit strategy as earlier i.e. $W_{exploit}$ = 1 in Eqn. 3.2. However, the key difference is that, it first sets value for last stored target position ($lstp$) variable. The value for $lstp$ is set as closest target position in the sensor range before starting the Exploit action and then it starts executing the Exploit

steps. If at the start of update step ($us(i)$) no targets are present, the observer performs an Explore action by setting its destination as the position in $lstp$ variable (set at $us(i-1)$). It then updates the $lstp$ variable for $us(i)$ as $null$. In the next update step ($us(i+1)$) if the observer still does not have a target in range, $lstp$ will be $null$ and hence performs a random explore (true for any step where $lstp$ is $null$). As earlier, random explore sets destination as random point within one quarter of environment width and height centered on it.

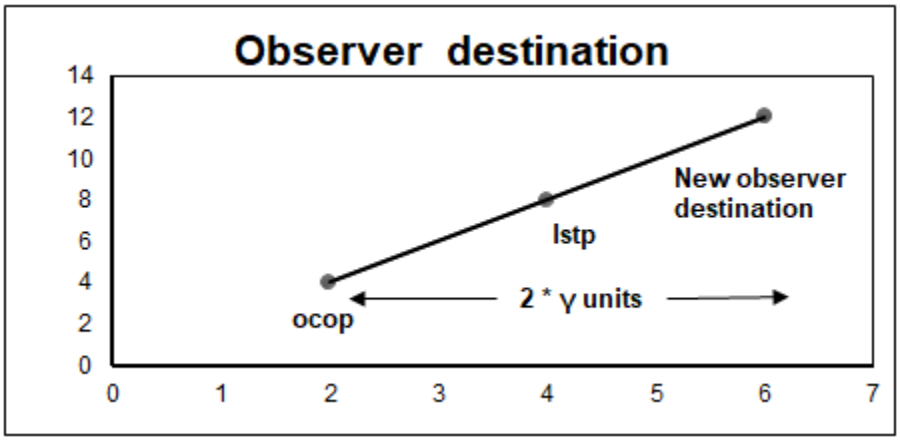

Figure 3.1 Calculating observer destination for One Step prediction

3.1.4 One Step Prediction

In the Memorization strategy, we try to capture the target which escaped observer's observation range by memorizing its last position and moving the observer to this last stored target position ($lstp$) with the hope of capturing the lost target again. We improve the memorization strategy by predicting the lost target's present position and move the observer to this position. If a target is present in observer's observation range, it performs the Exploit action from Explore-Exploit strategy [Eqn. 3.2].

One Step Prediction has a similar structure as Memorization except that it performs a prediction step. The key difference due to this happens in the Explore action wherein instead of setting destination (at $us(i)$) as the last stored target position ($lstp$), if no target is present in observer's range, we perform a prediction as where the target would be (at end of us(i)) using the $lstp$. To do this, we use the $lstp$ variable and *ocop* variable (observer's current origin position i.e. before taking first step at $us(i)$). Given that $2 * \gamma$ time-steps would have elapsed by the time the observer reaches its destination set in $us(i)$ (counting from when $lstp$ would be set, which is at $us(i-1)$), we predict the expected distance traveled would be time $(2 * \gamma) *$ target speed (1 unit/time-step) = $2 * \gamma$ units. We assume that target would travel the computed distance on the *ocop* to $lstp$ line and the expected target position is set as the destination for observer. After reaching the destination, if the observer still does not find any target in range, it sets $lstp$ to $null$. If $lstp$ is $null$ at the start of an Explore action, the target does random explore as in

Memorization by setting as destination, a random point within one quarter of environment width and height centered on it.

Fig. 3.1 shows the one step prediction for observer's destination. Observer's destination is set on a line segment joining $ocop$ (observer's current origin position) and $lstp$ (last stored target position). The distance between $ocop$ and observer's new destination will be $2 * \gamma$ units.

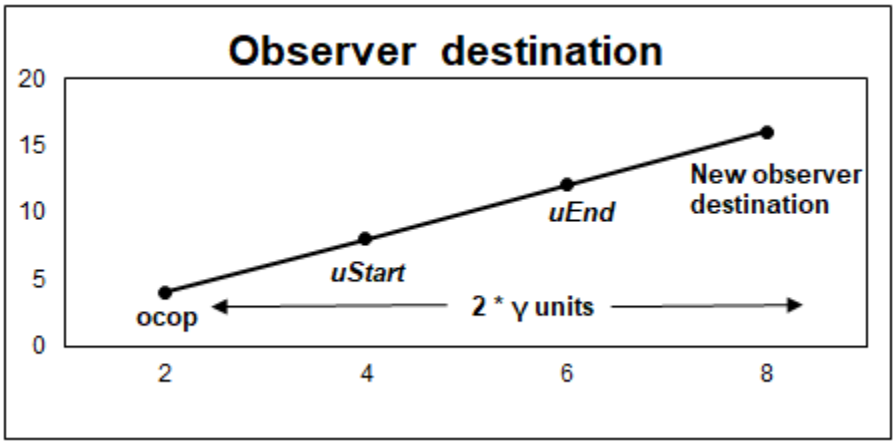

Figure 3.2 Calculating observer destination for k Step prediction

3.1.5 k-step Prediction

k-step Prediction has similar structure as One Step Prediction but makes k prior observations of the target instead of one $lstp$ variable. As earlier, at the start of every update step (say $us(i)$), each observer finds targets in its sensor range. If targets are present it performs an Exploit action. The key difference arises in the data collected: The observer collects ID of the nearest target (say target t_j) before taking the first step of Exploit action in $us(i)$. Collecting the ID allows the observer to track the same target afterwards (and avoids mix up of different targets and their locations). Thereafter for each of the next γ steps of $us(i)$, the observer maintains the location of target t_j. If t_j is out of sensor range, it stores $null$ for that step. It then identifies the start and end positions when t_j was in continuous observation, called the $uStart$ and $uEnd$. At the end of $us(i)$ update step, if there is no target in observer's observation range it predicts target position. We assume the target to be moving away from $uStart$ in a straight line connecting $uStart$ to $uEnd$. Given that γ time-steps elapse by the time observer reaches its destination in $us(i)$, we predict the distance traveled by the target from $uEnd$ as time multiplied by speed of target, i.e., $(\gamma - uEnd + \gamma) *$ target speed (1 unit/time-step) = $(2 * \gamma - uEnd)$ units. We set this expected position of target as the observer destination (for $us(i)$). After reaching the destination, if the observer still does not find any target in range, it sets all the location values to $null$. If no prior target information is available or if $uStart$ equals $uEnd$ for $us(i-1)$ at the start of an Explore action, the target does random explore as in Memorization.

Fig. 3.2 shows the k step prediction for observer's destination. Observer's destination is set on a line segment joining $ocop$ (observer's current origin position) with $uStart$ and $uEnd$. The distance from $ocop$ to observer's new destination will be $2 * \gamma$ units as 2 update steps would have elapsed by then.

3.1.6 Explore-Exploit with Adjustable Randomization

One of the key contributions of this work is to introduce controlled randomization into the observer strategy to make it less predictable. As shown in experiments section, the Explore-Exploit strategy is found to be the best strategy for observer across a variety of settings. Hence we pick Explore-Exploit as the strategy into which we introduce adjustable randomization. Intentionally introducing randomness into agent strategy has been used as a technique in the literature [7] to make the agent strategy less predictable.

We adapt the BRLP algorithm [36] introduced in the context of randomization in MDP based planning to our setting. To adapt this into our solution, we first define the notion of reward for observer in a CTO problem as the mean number of targets being observed by the observer in one simulation run (which as mentioned in our experimental setup has been set as 1500 time-steps).

Randomness in observer strategy will be computed over the weighing factor (i.e. α variable) in Eqn. 3.1. In the adjustable randomization strategy, α value is discretized into units of 0.1 varying from 0.1 to 1. A randomized solution would suggest a probability distribution over α vector instead of picking a specific α value that will maximize the observer expected reward i.e. maximize the expected number of targets observed. Randomness in strategy is measured using entropy. However an optimization program involving entropy as the objective function is non-linear in nature and will be intractable. The BRLP algorithm [36] introduced in the context of randomization in MDP based planning (BRLP-MDP), is a heuristic developed to address the problem related to non-linearity of the entropy function. We adapt the BRLP-MDP algorithm for our purposes here to develop the BRLP-CTO algorithm.

At each update step of the Explore-Exploit algorithm, the observer needs to compute the optimal α to use. It therefore calls BRLP-CTO algorithm to return the probability distribution over α and one particular value of α is picked based on the distribution returned. A key step involved in BRLP-CTO is the computation of the following Linear Program Eqn. (3.3). This LP computes the optimal probability distribution over α i.e. $p(\alpha)$ given a $r(\alpha)$ vector, a template probability distribution vector $p(\bar{\alpha})$ and a value for β as input to the LP. The template probability vector can be any distribution that has a high entropy and is useful to enforce some level of randomness into the solution since the objective function is reward maximization. One such high entropy probability distribution vector is the uniform distribution vector. The amount of randomness that is reflected in the solution from the template vector is controlled by $\beta \in [0, 1]$.

The core issue with adapting BRLP-MDP [36] to the CTO problem is with modeling of $r(\alpha)$. Unlike an MDP where the reward function is input to the problem, we need to perform a real-time simulation over all the anticipated scenarios to construct the reward function. Following are the steps involved in this construction:

1. Each observer $o \in O$ notes the position, speed and direction of all targets $t_o \in T_o$ in its sensor range over the γ time-steps in current update step ($us(i)$). Each target added to T_o may be in sensor range at different time-steps and durations as long as they fall within the window from end of first step in $us(i)$ to the start of first step in $us(i+1)$.

2. Using information from the γ sensing steps, at the start of $us(i+1)$ all the observers $o \in O$ estimate the destination position $dp^o_{t_o}$ for each $t_o \in T_o$ at start of $us(i+2)$. This is called an estimation step and can take place only at the start of update step.

3. Using the estimated values, at the start of $us(i+1)$, every observer o computes the estimated mean position of targets that will lie in its sensor range at $us(i+2)$. Then, every observer o for each α value uses Eqn. 3.2 (with $W_{explore}$ = 0) to compute its own possible destination position dp^o_α and checks whether each of the targets in $dp^o_{t_o}$ falls within the sensing range of dp^o_α.

4. For each T_o that o predicts will be in sensing range of dp^o_α at $us(i+2)$, it adds $+1$ to its reward at α. Hence the reward vector $r(\alpha)$ for $us(i+1)$ gets estimated.

We now present the LP (3.3) which takes β value as input (which as we will see the BRLP-CTO provides):

$$\begin{aligned} &\text{Maximize} \sum_{\alpha=0.1}^{1.0} p(\alpha) r(\alpha) \\ &s.t. \sum_{\alpha=0.1}^{1.0} p(\alpha) = 1, \forall p(\alpha) >= \beta p(\bar{\alpha}) \end{aligned} \tag{3.3}$$

BRLP-CTO identifies the appropriate β value by doing the following: It performs a binary search over the β space to attain a policy with expected reward $E(\beta)$ which lies between $\{\bar{E}, E^*\}$, by adjusting the parameter β. For $\beta = 0$ this problem reduces to a LP without any constraint on $p(\alpha)$ and hence returns the highest expected reward of E^*. For $\beta = 1$ it returns the template policy $p(\bar{\alpha})$ which acts as a lower bound on the expected reward ($\bar{E}$). The binary search exploits this inverse relationship between β and $E(\beta)$ to identify the appropriate β (and hence $p(\alpha)_\beta$) that guarantees the threshold amount of reward E_{min} that a user requests for. BRLP-CTO has fast convergence (average in milliseconds), hence it is suited for dynamic environments.

Algorithm 1: BRLP-CTO (E_{min}, $p(\bar{\alpha})$)

Set $\beta_l = 0$, $\beta_u = 1$ and $\beta = 1/2$.;
Solve Problem (3.3), let $p(\alpha)_\beta$ and $E(\beta)$ be the optimal solution and expected reward value obtained.;
if $E_{min} > \bar{E}$ **then**
    while $|E(\beta) - E_{min}| > \epsilon$ **do**
        if $E(\beta) > E_{min}$ **then**
            Set $\beta_l = \beta$;
        else
            Set $\beta_u = \beta$;
        end
        $\beta = \frac{\beta_l + \beta_u}{2}$;
        Solve Problem (3.3), let $p(\alpha)_\beta$ and $E(\beta)$ be the optimal solution and expected reward value returned;
    end
end
return $p(\alpha)_\beta$;

3.2 Targets Movement

Prior work in the literature assumes that targets move randomly. However in a surveillance domain it is reasonable to assume that targets would be strategic and try to minimize their observation. Targets have a sensor range similar to observers in which they can see observers. Like for observers, we define targets movement by setting a destination point and moving a fixed γ_{target} number of steps (with default value of 100, note that it was γ steps for observers) towards it. If the target reaches its destination position before γ_{target} steps, it immediately calculates its new destination similar to the target movement in [23]. Targets destination position is computed by using following algorithms:

- **Randomized Movement**
- **Straight-line Movement**
- **Controlled Randomization**

3.2.1 Randomized Movement

In the Randomized Movement, target destination position is chosen at random from within a local region (one quarter of the environment height and width) centered on the target as it has a local view of environment. Target then moves to the destination point for utmost γ_{target} time-steps whose default

value is set as 100. If it reaches its destination before γ_{target} time-steps, it calculates its new destination by again selecting a random point from one quarter of environment width and height.

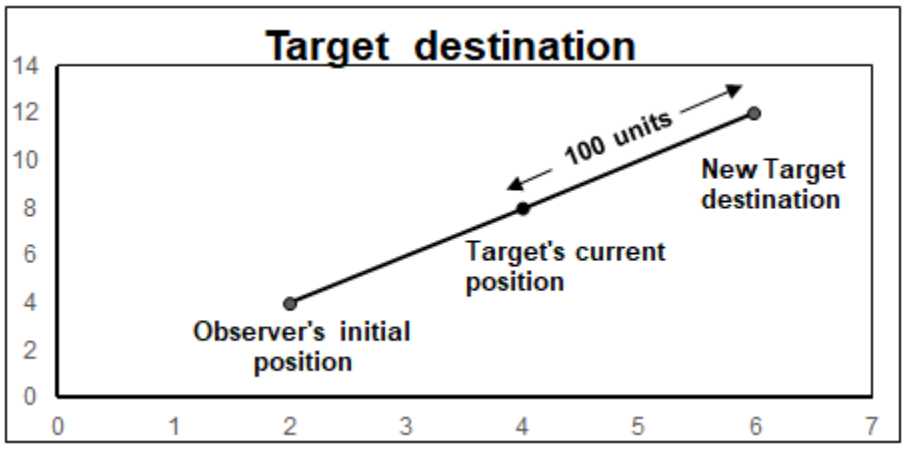

Figure 3.3 Calculating target destination for Straight-line Movement

3.2.2 Straight-line Movement

Our first heuristic namely Straight-line Movement wherein a target attempts to escape out of the observer's observation range by following a straight line trajectory in the direction that maximizes the distance of the target from the observer. This direction is computed by joining a line segment between the observer's current position and the target in the direction of target. Target takes γ_{target} steps in this direction after which it again calculates the new destination depending on the observer observing it then. If there is more than one observer observing the target, it calculates the mean position of observers and uses it to identify the direction which leads to the maximum distance from the mean.

Fig. 3.3 shows the straight-line path followed by the target for 100 time-steps in the direction of line segment joining observer's initial position and target's current position.

3.2.3 Controlled Randomization

The Straight-line Movement has a downfall that it might increase observer's suspicion towards target. To reduce suspicion, we propose Controlled Randomization which adds randomization to the Straight-line Movement. Randomization decreases the suspicion towards a target as the observer cannot identify any specific goal for target while Straight-line Movement helps target to escape out of the sensor range of observer by creating maximum distance from it. In Controlled Randomization, we perform a combination of both the strategies, to decrease its suspicion and also to escape out of observer's observation range. The targets follow the Straight-line Movement for η time-steps and Randomized Movement for the remaining time-steps. We experimented with various values for η for example 0.7, 0.5, 0.3 and a

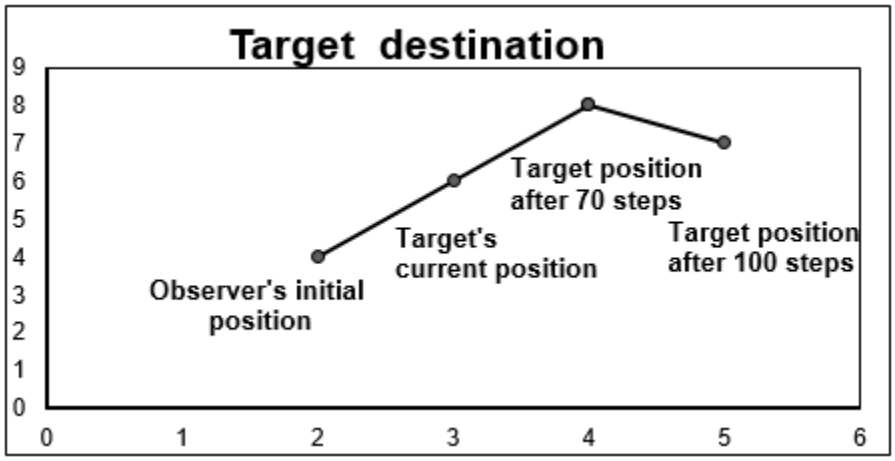

Figure 3.4 Calculating target destination for Controlled Randomization Movement

value of 0.7 was found to perform the best. Hence, the target moves in the direction of destination for .7 * γ_{target} time-steps = 70 time-steps and follows a random movement henceforth for 30 time-steps.

Fig. 3.4 represents the Controlled Randomization path followed by target. It moves in a straight-line fashion away from observer's initial position on a line segment joining the observer and target for 70 time-steps and then moves towards a Random point for remaining 30 time-steps.

3.3 Summary

We propose strategies for observer and target agents. We set the observer's next destination taking into account the limited knowledge of the environment which the observer has, i.e. the previous positions of the target agents present in the observation range of the observer. Observer tries to predict the target's position once the target moves out of the observer's range and moves to this position in search of the target. We also adapt the BRLP algorithm introduced in the context of randomization in MDP based planning to our setting to make the observer's action less predictable. Target strategies are based on the real world scenario where the target would attempt to maximize its distance from observer in order to reduce its observation. In next Chapter, we will discuss the comparison of these strategies and present the best strategy for agents.

Chapter 4

Experiments

4.1 Parameter description

For purposes of experiments, we assume that the observers and targets are operating in a rectangular field with a width and height of 150 x 150 units (we refer to as units for generalization), though the area of interest can be of any shape, it can be finite or infinite for example, in case of border patrolling the area of interest is infinite. We performed a variety of experiments by varying the density, speed and sensor range of agents, where density is a function of number of observers and targets as the area is the same for all.

- To vary density, the number of observers was picked from $\{2, 6, 10, 14, 18\}$ and the number of targets from $\{3, 9, 15, 21, 27\}$. We performed experiments by varying the number of observers and targets together.
- Six possible target speed values were picked among $\{0.2, 0.5, 0.8, 1.0, 1.2, 1.5\}$ measured as units per time-step. The speed of observers was fixed to 1 unit per time-step.
- Five possible sensor range values picked from $\{5, 10, 15, 20, 25\}$ where a value 5 represents 5 units.
- In total there are 5*5 i.e. 25 settings for densities. They were tested against six possible values for target speeds and five different values for sensor ranges resulting in a test bed involving 5*5*6*5 = 750 different settings.
- We also have different algorithms for calculating observer and target destination positions which were experimented against each other. We compared 5 algorithms for observer and 3 algorithms for targets against each other.
- Each experiment was simulated 30 times (number of runs) and each simulation run consists of a total of 1500 time-steps i.e. each observer and target takes 1500 steps (can stay at same position or move in a step).

- While performing the experiments, for each simulation run we gathered the number of time-steps each target is under observation. The mean across the 30 runs gives the mean number of steps each target is under observation per experiment. If a target was observed by multiple observers, we counted the target only once.

- We use mean number of targets observed as the criteria for deciding the best strategy for observer and target. The best strategy for observer will be the one in which the mean number of targets observed will be maximum as observer's aim is to maximize its surveillance. Targets best strategy will be the one in which the mean number of targets happens to be minimum as target's aim is to minimize its observation, escape from observer's observation.

4.2 Tools and Libraries

4.2.1 MASON: A Simulation Software

All our experiments were performed on MASON simulation toolkit [22]. MASON is a fast and discrete-event multi-agent simulation library. We implemented our model over MASON and used its internal threading mechanism to run the agents in parallel. MASON has a layered architecture: Model layer and Visualization layer. Model can be detached from the visualization layer and run without a visualizer. It can also be attached to a different GUI toolkit for visualization.

4.2.1.1 Model Layer

A single instance of the SimState class contains the MASON model. SimState class is a subclass of MASON model class. This instance has a discrete-event schedule which employs a specific usage of agent. Agents are modeled to follow specific behavior and they take actions accordingly. An agent can be scheduled to perform different functions multiple times using a wrapper class. The wrapper can group the agents together and run them in parallel on different threads. We use this wrapper to group our agents into two categories: Targets and Observers, and we have the two groups run in parallel with the agents in each group running parallel too.

4.2.1.2 Visualization Layer

SimState class has a wrapper called GUIState which separates visualization layer from model layer. GUIState class can separate the model entirely from visualizer and serialize SimState class to or from disk. GUIState has its own mini-schedule to schedule visualization windows which need to be scheduled to reflect the changes of the model. This allows model class to function without the visualizer.

4.3 MASON Simulation

Below are the screenshots of three consecutive steps in our simulation:

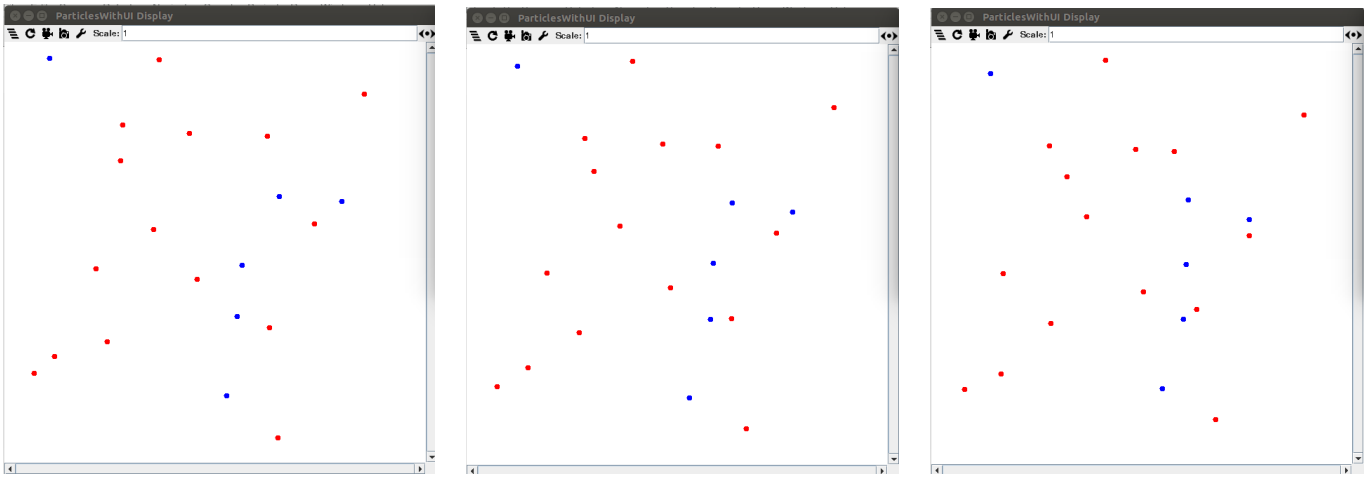

Figure 4.1 CTO Simulation

In Fig. 4.1, we have 6 observers, 15 targets, sensor range of the observer is 15 units and target speed is 1.0 units/sec. Observers are the blue colored agents and targets are the red colored agents. Here, the observers follow Explore-Exploit strategy and targets follow Randomized strategy. We can see from the series of simulations that the observer attempts to keep the target agent in its range by setting its destination closer to the mean of the targets in its sensor range.

4.4 Experiments

4.4.1 Optimal α value

Our first experiment identifies the best α value for observers using K-means algorithm. α is the parameter which weighs appropriately the observer's initial position and the mean of the position of targets in the observation range of observer to compute the observer's new destination. For this experiment, experimental parameters are set as follows: number of observers as 18, number of targets as $\{3, 9, 15\}$, sensor range as 15, targets follow Randomized strategy for movement, observers follow K-means based algorithm, α is discretized using interval of 0.1 from 0.1 to 1 and obtained the following vector of average number of targets observed: <4.39, 4.7, 5.57, 6.52, 6.69, 6.63, 6.66, 6.69, 6.73, 6.82> (corresponding to each value of alpha from 0.1 to 1). We can see that the average number of targets is maximum for α = 1 which implies maximum targets are observed when we select α as 1. Henceforth, we set α to 1 in all our experiments since it performs the best. Prior work on k-means [23] used α as 0.25 which we found to be significantly worse for our setup as the aim in [23] was not to maximize the

observer's observation but to study the effect of decentralization on two algorithms namely K-means clustering and hill-climbing.

Model	3	9	15	21	27
Model 1	2.3	5.48	7.77	9.72	11.47
Model 2	1.76	4.58	6.94	8.96	10.9
Model 3	2.21	5.28	7.65	9.51	11.41

Table 4.1 Comparison of Explore-Exploit models

4.4.2 Explore-Exploit Model

The second experiment identifies the best weight vector to use for the Explore and Exploit components in Observers Explore-Exploit strategy. The Explore component of this strategy explores the area of interest in search of targets when there is no target in observer's observation range and Exploit component helps in observing the targets present in observation range. We need a combination of both the components. Table 4.1 compares the performance measured as the mean number of targets observed per experiment for the three Explore-Exploit models. Each row of the table corresponds to a model and the columns represent the mean number of targets observed with this Model. For purposes of this experiment we set the experimental parameters as: number of observers as 10, target speed as 1.0, sensor range as 15 and targets following Randomized strategy. Based on these experiments, Model 0-1 is found to perform consistently better than the rest, as the maximum mean number of targets are observed with the Model 0-1 on an average over 30 runs with 1500 time-steps in each run. Hence, we use weight vectors defined in Model 0-1 for further experimentation.

4.4.3 Mixing Ratio for Controlled Randomization

This experiment aims to identify the best ratio of Straight-line and Randomized steps for Controlled Randomization strategy. We tested three different mixing ratios: 50-50, 70-30 and 30-70, where 30-70 implies 30% steps in Straight-line and 70% Random. Table 4.2 illustrates the results measured as the mean number of targets observed. Each row of the table corresponds to one particular mixing ratio while the columns correspond to number of targets. Other parameters were set as follows: number of observers as 10, target speed as 1.0, sensor range as 15 and observers following Explore-Exploit strategy. The table 4.2 shows that the minimum number of targets are observed when targets use a 70:30 mixing ratio and hence we set 70:30 as the default mixing ratio for Controlled Randomization strategy.

Proportions	3	9	15	21	27
70-30	2.57	5.67	7.88	9.97	11.62
50-50	2.62	6	8.11	10.03	11.93
30-70	2.6	5.83	7.93	10.07	11.69

Table 4.2 Comparison of Controlled Randomization ratios

4.4.4 Target Strategies Comparison

In this experiment, we study the performance of Straight-line strategy and Controlled Randomization strategy (with 70:30 mix) and compare these with Randomized strategy. Our study shows a significant decrease in the mean number of targets observed due to their ability to avoid observer. The experiments were performed using a setting of sensor range as 15, number of observers as $\{2, 6, 10, 14, 18\}$, number of targets as $\{3, 9, 15, 21, 27\}$, target speeds as $\{0.5, 0.8, 1.0, 1.2\}$ and observers following Explore-Exploit strategy. The mean number of targets observed by a strategy is calculated by averaging the number of targets observed in each experiment. The number of experiments performed for each target strategy are: 5*5*4 = 100. Each experiment consists of 30 simulation runs of 1500 steps each.

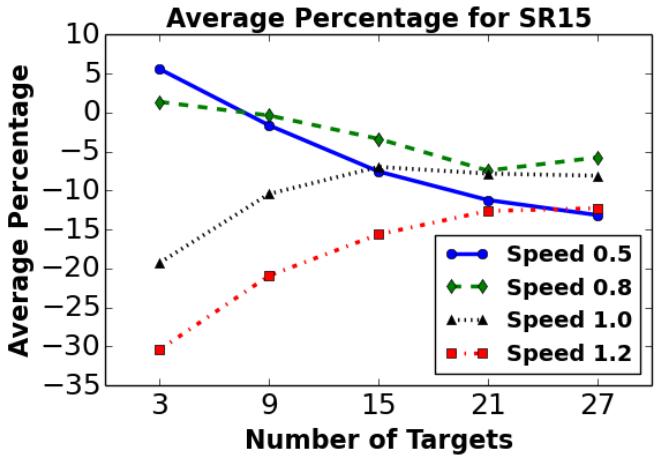

Figure 4.2 Straight-line vs Random

The percentage difference between strategies in terms of mean number of targets observed is calculated as $\frac{T_a - T_b}{T_b} \times 100$, where T_a denotes mean number of targets observed by strategy a and T_b denotes mean number of targets observed by strategy b.

Fig. 4.2 shows the number of targets on X-axis and depicts the percentage difference in the mean number of targets observed by Straight-line (strategy a) with respect to the Randomized (strategy b) on Y-axis. The lines of the graph represent the percentage difference between the strategies. In Fig. 4.2, we can see that the lines are completely in the negative Y-axis which implies that the mean number of

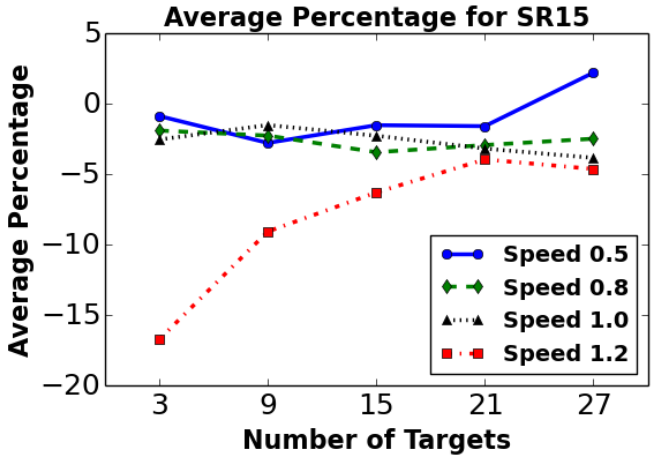

Figure 4.3 Controlled Random vs Random

targets observed in Straight-line strategy is less than Randomized strategy. This shows that the targets are able to avoid the observer with Straight-line strategy. The 4 lines correspond to the 4 different target speed settings indicated. The Figure also shows a trend wherein as the target speeds get higher from 0.5 to 1.2 there is a marked decrease in the percentage difference which shows that the Straight-line strategy at higher target speeds starts becoming more effective.

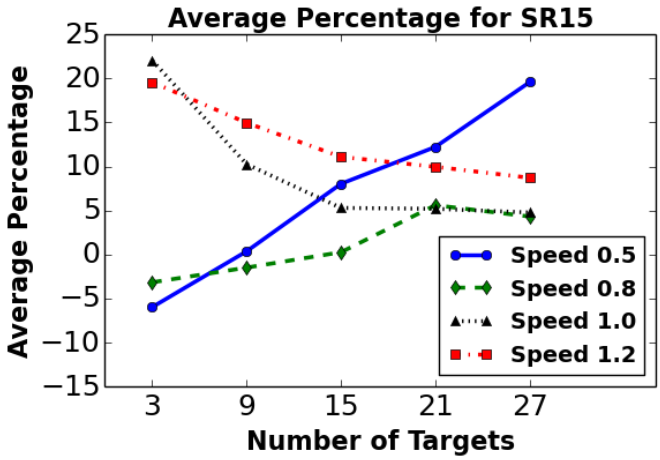

Figure 4.4 Controlled Random vs Straight-line

Fig. 4.3 shows the percentage difference in the mean number of targets observed by targets following Controlled Randomization (strategy a) having 30% randomization with respect to the Randomized (strategy b). Similar to Fig. 4.2, all the lines representing the percentage difference between the strategies are in the negative Y-axis. While the magnitude of negative value is lower than Fig. 4.2 on an average, it implies that targets are able to overcome observer's observation radius easily in Straight-

line strategy as compared to Controlled Randomization. The four lines indicate that as the target speed increases and becomes greater than observers speed, the Controlled Randomization strategy becomes more effective.

Fig. 4.4 compares the Controlled Randomization strategy (strategy a) with Straight-line strategy (strategy b). Most points lie on the positive Y-axis which implies Straight-line strategy is more effective than Controlled Randomization. This is because the random part of the path decreases the distance between observer and target and leads to increase in target observation. While we leave mathematical quantification of suspicion for future work, our goal of introducing randomness is to reduce the suspicion that target is always trying to move away and will be used as default strategy for targets.

4.4.5 Observer Strategies Comparison

In this experiment, we study the performance of Memorization, One Step and k-step Prediction in comparison to the Explore-Exploit strategy. The experiments were performed using a setting having a sensor range as 15, number of observers as $\{2, 6, 10, 14, 18\}$, number of targets as $\{3, 9, 15, 21, 27\}$ and target speed as $\{0.5, 0.8, 1.0, 1.2\}$. Targets were modeled to follow Controlled Randomization strategy. The X-axis of Fig. 4.5, Fig. 4.6 and Fig. 4.7 shows the number of targets while the Y-axis shows the percentage difference in the mean number of targets being observed by two strategies. The four lines correspond to four different target speed settings corresponding to $\{0.5, 0.8, 1.0, 1.2\}$. As earlier, each point in the graph is an average over 30 runs (30*1500 time-steps) across all the settings for number of observers.

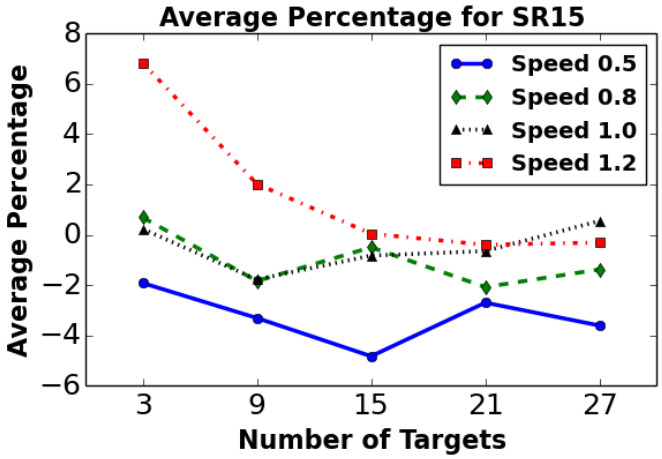

Figure 4.5 Memorization vs Explore-Exploit

4.4.5.1 Memorization vs Explore-Exploit strategy

Fig. 4.5 compares the Memorization strategy (strategy a) against Explore-Exploit (strategy b). As we can see from the Figure, for the highest speed setting almost all the points lie on positive Y-axis implying Memorization outperforms Explore-Exploit. However, apart from the highest speed setting Explore-Exploit outperforms for all the other settings. The lower performance for Memorization is possibly due to the fact that the observer is only moving to the last position of the nearest target and not searching for other targets. Fig. 4.5 also shows that Memorization performs better for lower target density. This is because in case of low target density, observer attempts to catch the lost target from its observation range as there are less targets in the area, whereas in the case of high target density, there is a higher chance that Explore action would result in more number of targets in observers range.

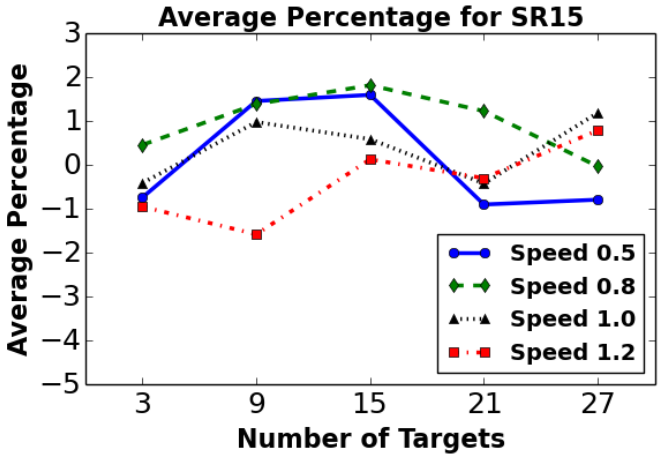

Figure 4.6 One Step vs Explore-Exploit

4.4.5.2 One step prediction vs Explore-Exploit strategy

In Fig. 4.6, we compare One Step Prediction against Explore-Exploit strategy. The Figure shows that there is no clear winner among Explore-Exploit and One Step Prediction, though One step performs better than Explore-Exploit for low target speeds and Explore-Exploit performs better at high target speeds. As for low target speeds, target is less likely to move far from the observer, hence observer has a high chance of reaching the target with the One-step as in this strategy we predict the current position of target and observer sets this position as its destination. In the case of high target speed, target would have moved far away from the observer, hence random exploration of the environment in Explore-Exploit strategy performs better. One step also performs better when the target density is low as compared to high target density, as in case of less number of targets, it is better to try to find the lost target than explore the area in search of targets.

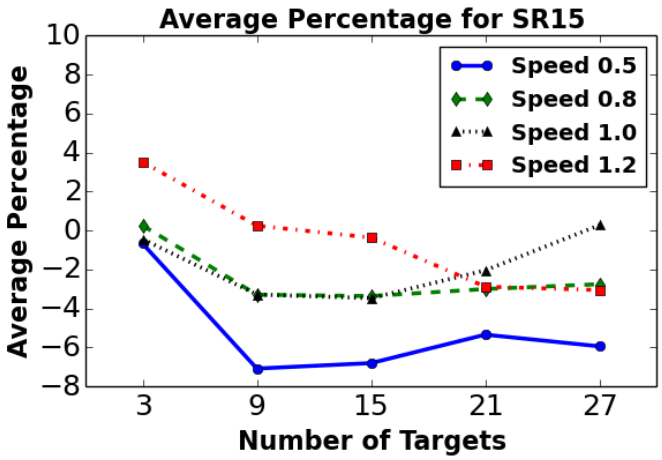

Figure 4.7 k-Step vs Explore-Exploit

4.4.5.3 k step prediction vs Explore-Exploit strategy

In Fig. 4.7, we compare k step Prediction against Explore-Exploit strategy. Explore-Exploit performs better than k step for almost all settings, though k step performs better when the target density is low. This happens due to the same reason as in case of One Step prediction. When the number of targets is less, observer tries to follow the last nearest target in case where no target is in range instead of searching for more targets. When target density increases, the prediction does not fetch many targets to the observer as compared to Explore-Exploit. The 0-1 Explore-Exploit will therefore be used as the default observer strategy.

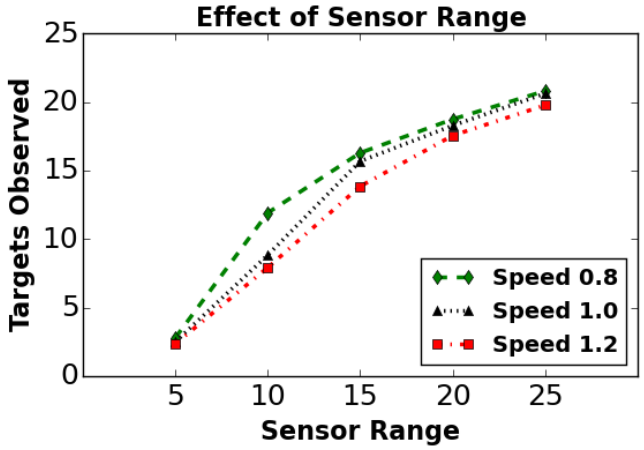

Figure 4.8 Effect of sensor range on mean number of targets observed

4.4.6 Effect of sensor range on mean number of targets observed

In this experiment, we study the effect of sensor range on average number of targets observed. The experiment was performed with setting having sensor range as $\{5, 10, 15, 20, 25\}$, number of observers as 18, number of targets as 27 and target speed as $\{0.8, 1.0, 1.2\}$. We keep the density of observers and targets as constant to study the effect of sensor range alone. Observers are modeled to follow Explore-Exploit strategy and targets are modelled to follow Controlled Randomization strategy for this experiment. We run one experiment setting for 30 iterations and take the average of the targets observed in 30 iterations as mean number of targets observed.

Fig. 4.8 depicts observer's sensor range as X-axis and mean number of targets observed with this sensor range as Y-axis. It shows that with the increase of sensor range, the mean number of targets observed by the observer also increases. This happens due to the increase in the area covered by the observer in one simulation run and as this area increases more number of targets can be observed by the observer. Also, the Figure shows that more number of targets are being observed at low target speed.

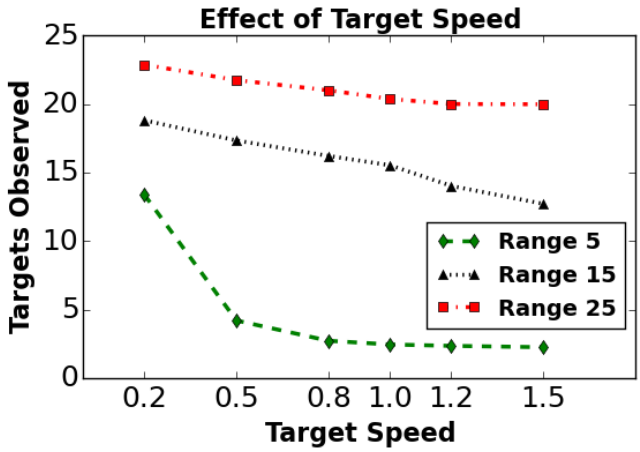

Figure 4.9 Effect of target speed on mean number of targets observed

4.4.7 Effect of target speed on mean number of targets observed

In this experiment, we study the effect of target speed on the average number of targets observed. The experiment was performed using a setting of sensor range as $\{5, 15, 25\}$, number of observers as 18, number of targets as 27 and target speed as $\{0.2, 0.5, 0.8, 1.0, 1.2, 1.5\}$, we keep the observer and target density constant to study the effect of target speed alone. Similar to the above experiment, Observers are modeled to follow Explore-Exploit strategy and targets are modeled to follow Controlled Randomization strategy. We run one experiment setting for 30 iterations and take the average of the targets observed in 30 iterations.

Fig. 4.9 represents the target speed on X-axis and mean number of targets observed across the 30 iterations for this target speed on Y-axis. It shows that with the increase of target speed, the mean number of targets observed decreases as with the increase in target speed, targets are able to come out of observers observation radius. We observed a similar pattern for target speed in the above experiment as well. Mean number of targets observed increases with the increase in sensor range which also holds true as per our previous experiment.

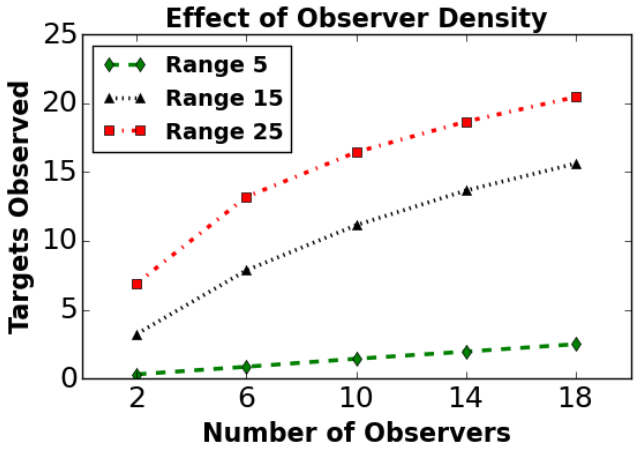

Figure 4.10 Effect of observer density on mean number of targets observed

4.4.8 Effect of observer density on mean number of targets observed

In this experiment, we study the effect of observer density on the number of targets observed. The experiment was performed using a setting of sensor range as $\{5, 15, 25\}$, number of observers as $\{2, 6, 10, 14, 18\}$, number of targets as 27 and target speed as 1.0, we keep the target speed and target density constant to study the effect of observer density alone. Similar to the above experiments, Observers are modelled to follow Explore-Exploit strategy and targets are modelled to follow Controlled Randomization strategy. We run one experiment setting for 30 iterations and take the average of the targets observed in 30 iterations.

Fig. 4.10 represents the number of observers on X-axis and mean number of targets observed across the 30 iterations for this observer density on Y-axis. We can see with the increase in number of observers, mean number of targets also increase. This result is intuitive as with the increase in number of observers and constant number of targets, more observers can observe the targets. Hence, mean number of targets observed increases. As with earlier experiment with increase of sensor range, mean number of targets also increases holds true in this experiment as well.

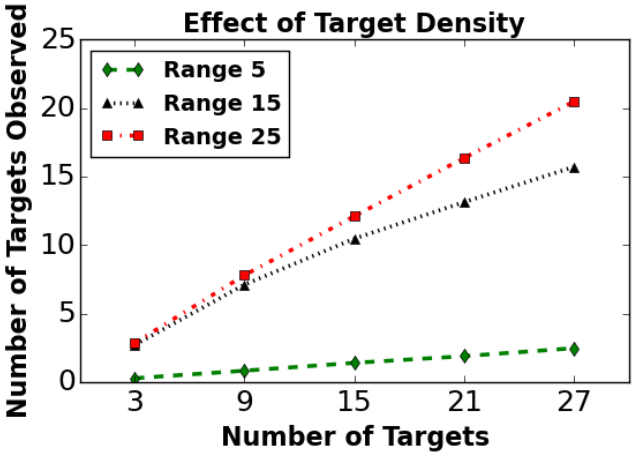

Figure 4.11 Effect of target density on mean number of targets observed

4.4.9 Effect of target density on mean number of targets observed

In this experiment, we study the effect of target density on the number of targets observed. The experiment was performed using a setting of sensor range as $\{5, 15, 25\}$, number of observers as 18 number of targets as $\{3, 9, 15, 21, 27\}$ and target speed as 1.0. We keep the target speed and observer density constant to study the effect of target density alone. Similar to the above experiments, observers follow Explore-Exploit strategy and targets follow Controlled Randomization strategy. We run one experiment setting for 30 iterations and take the average of the targets observed in 30 iterations.

Fig. 4.11 represents the number of targets on X-axis and mean number of targets observed across the 30 iterations for this target density on Y-axis. We can see with the increase in number of targets, mean number of targets also increase. This result is also intuitive as with the increase in number of targets and constant number of observers, the observers can observe more targets. Hence, mean number of targets observed increases. As with earlier experiment with increase of sensor range, mean number of targets also increases holds true in this experiment as well.

4.4.10 Reward vs. Entropy Trade off for BRLP-CTO

For this experiment, we vary the percentage of optimal reward (i.e. threshold) that the user would like to obtain using the BRLP-CTO to increase randomization. We performed this experiment with sensor range as 15, target speed as $\{0.8, 1.2\}$, number of observers as $\{2, 10, 18\}$, number of targets as $\{3, 15, 27\}$ and targets following Controlled Randomization. We obtained the following vector of average entropy values (averaged across all the settings for speeds, number of observers, targets and runs): $<$3.321, 3.319, 3.312, 3.26, 3.041, 2.137$>$ for threshold reward ranging from 50% to 100% in increments of 10%. Note that as expected it is a entropy vector with decreasing values with 100% threshold giving the lowest entropy.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In this thesis, we adapt the Cooperative Target Observation (CTO) problem to Surveillance domain. We have two sets of agents in our problem domain namely Observers and targets. The agents have a limited visibility in terms of sensor range with a 360 ° view of the environment. There is no communication involved among them. The aim of observers is to collectively attempt to keep maximum number of targets within their observation range. The aim of targets is to minimize their observation as much as possible.

In related literature, K-means clustering algorithm is used for observers movement. We improved upon the K-means algorithm to develop five variant strategies for the observer namely, Explore-Exploit, Memorization, One-step Prediction, k-step Prediction and Explore-Exploit with Adjustable Randomization. These strategies are created taking into account the observer's current position and target's previous positions. Observer uses these strategies to calculate its destination for next step in order to maximize its observation.

We changed the assumption related to targets to make them strategic (as opposed to following a Randomized strategy in prior works) via development of two heuristics namely, Straight-line Movement and Controlled Randomization which would help in minimizing their observation. These strategies were found in our experiments to make better decisions than Randomized approach. In Straight-line strategy, target always moves away from the observer observing it which in turn increases its suspicion from observer. We tried to counter this issue with Controlled Randomization strategy, in which targets follow Straight-line strategy for 70 steps and then take a Randomized approach for next 30 steps.

We simulated the system using MASON simulation software with targets and observers running in parallel threads and executing their behaviors. We performed a series of experiments to show that the 0-1 Explore-Exploit strategy performs best for the observers. We built upon this strategy to introduce user adjustable randomization into the surveillance plan so that the observer agent could not be harmed by some adversary. By increasing the entropy of the system, we are decreasing the predictability of the observer's actions otherwise an adversary can study the movement of the observer and predict its

future movement. In order to achieve the increase in entropy, we are sacrificing our reward i.e. the mean number of targets observed. We presented the trade-off between Reward Vs Entropy as one of our experiments.

To summarize, we studied the effect of various parameters affecting the movement of observers and targets such as, sensor range, target speed, target and observer density in our experiments. We observed that with increase in sensor range of observer, it is able to observe more targets as the visible area under its observation increases. Our experiments also show that mean number of targets observed increases when the targets move with low speed and also with increase in density of observers and targets.

Based on all the results obtained, we conclude that with constraints placed on communication, sensor range and action prediction for agents, Explore-Exploit with Adjustable Randomization turns out to be the best strategy for observers while Controlled Randomization turns out to be the best strategy for targets.

5.2 Future Work

We believe this work provides the needed base for using Cooperative Target Observation for improving surveillance and can be expanded in a variety of ways in future:

- We plan to add communication to our observer agents. Observers will be able to make better decisions with communication among them as they will have an overall view of the system, if one observer is observing a target, other observers will not attempt to observe that target.
- The second direction to future work is to study the inclusion of realistic constraints in the environment in which the agents operate like, obstacles and terrains which can restrict the observer's visibility.
- We also plan to develop more realistic and efficient algorithms for observers and targets.

Related Publications

- Aswani, Rashi, Sai Krishna Munnangi, and Praveen Paruchuri. ”Improving Surveillance Using Cooperative Target Observation.” In AAAI, pp. 2985-2991. 2017.

Bibliography

[1] M. Aigner and M. Fromme. A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1–12, 1984.

[2] L. Alonso, A. S. Goldstein, and E. M. Reingold. lion and man: Upper and lower bounds. *ORSA Journal on Computing*, 4(4):447–452, 1992.

[3] B. An, F. Ordóñez, M. Tambe, E. Shieh, R. Yang, C. Baldwin, J. DiRenzo III, K. Moretti, B. Maule, and G. Meyer. A deployed quantal response-based patrol planning system for the us coast guard. *Interfaces*, 43(5):400–420, 2013.

[4] B. An, M. Tambe, and A. Sinha. Stackelberg security games (ssg): Basics and application overview. *Improving Homeland Security Decisions. Cambridge University Press, forthcoming*, 2015.

[5] T. Başar and G. J. Olsder. *Dynamic noncooperative game theory*. SIAM, 1998.

[6] A. Bonato and R. J. Nowakowski. *The game of cops and robbers on graphs*, volume 61. American Mathematical Society Providence, 2011.

[7] B. D. Bryant and R. Miikkulainen. Evolving stochastic controller networks for intelligent game agents. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1007–1014. IEEE, 2006.

[8] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks foo. 51(12):1448–1453, 2002.

[9] T. H. Chung, G. A. Hollinger, and V. Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299, 2011.

[10] S. S. Dhillon, K. Chakrabarty, and S. S. Iyengar. Sensor placement for grid coverage under imprecise detections. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 2, pages 1581–1587. IEEE, 2002.

[11] H. Everett, G. Gilbreath, T. Heath-Pastore, and R. Laird. Controlling multiple security robots in a warehouse environment. In *NASA CONFERENCE PUBLICATION*, pages 93–93. NASA, 1994.

[12] M. Garzón, J. Valente, D. Zapata, and A. Barrientos. An aerial-ground robotic system for navigation and obstacle mapping in large outdoor areas. *Sensors*, 13(1):1247–1267, 2013.

[13] H. González-Baños. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 232–240. ACM, 2001.

[14] J. P. Hespanha, H. J. Kim, and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 3, pages 2432–2437. IEEE, 1999.

[15] J. Hu, L. Xie, K.-Y. Lum, and J. Xu. Multiagent information fusion and cooperative control in target search. *IEEE Transactions on Control Systems Technology*, 21(4):1223–1235, 2013.

[16] V. Isler, S. Kannan, and S. Khanna. Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5):875–884, 2005.

[17] S. Jacobi, C. Madrigal-Mora, E. León-Soto, and K. Fischer. Agentsteel: An agent-based online system for the planning and observation of steel production. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 114–119. ACM, 2005.

[18] B. Jung and G. S. Sukhatme. Cooperative multi-robot target tracking. In *Distributed Autonomous Robotic Systems 7*, pages 81–90. Springer, 2006.

[19] A. Kolling and S. Carpin. Cooperative observation of multiple moving targets: an algorithm and its formalization. *The International Journal of Robotics Research*, 26(9):935–953, 2007.

[20] A. Kolling and S. Carpin. Pursuit-evasion on trees by robot teams. *IEEE Transactions on Robotics*, 26(1):32–47, 2010.

[21] J. E. Littlewood. *A mathematician's miscellany*. Methuen London, 1953.

[22] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan. Mason: A new multi-agent simulation toolkit. In *Proceedings of the 2004 swarmfest workshop*, volume 8, page 44, 2004.

[23] S. Luke, K. Sullivan, L. Panait, and G. Balan. Tunably decentralized algorithms for cooperative target observation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 911–917. ACM, 2005.

[24] Microdrones. https://www.microdrones.com/en/appli cations/areas-of-application/monitoring/. 2016.

[25] P. J. Nahin. *Chases and escapes: the mathematics of pursuit and evasion*. Princeton University Press, 2012.

[26] R. Nowakowski and P. Winkler. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3):235–239, 1983.

[27] J. O'rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

[28] L. E. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE transactions on robotics and automation*, 14(2):220–240, 1998.

[29] L. E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation & Soft Computing*, 5(1):5–19, 1999.

[30] L. E. Parker. Distributed algorithms for multi-robot observation of multiple moving targets. *Autonomous robots*, 12(3):231–255, 2002.

[31] L. E. Parker and B. A. Emmons. Cooperative multi-robot observation of multiple moving targets. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2082–2089. IEEE, 1997.

[32] T. D. Parsons. Pursuit-evasion in a graph. In *Theory and applications of graphs*, pages 426–441. Springer, 1978.

[33] T. D. Parsons. The search number of a connected graph. In *Proc. 9th South-Eastern Conf. on Combinatorics, Graph Theory, and Computing*, pages 549–554, 1978.

[34] P. Paruchuri. *Keep the adversary guessing: Agent security by policy randomization*. University of Southern California, 2007.

[35] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordonez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[36] P. Paruchuri, M. Tambe, F. Ordóñez, and S. Kraus. Security in multiagent systems by policy randomization. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 273–280. ACM, 2006.

[37] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[38] M. M. Polycarpou, Y. Yang, and K. M. Passino. A cooperative search framework for distributed agents. In *Intelligent Control, 2001.(ISIC'01). Proceedings of the 2001 IEEE International Symposium on*, pages 1–6. IEEE, 2001.

[39] M. Popa, L. Rothkrantz, Z. Yang, P. Wiggers, R. Braspenning, and C. Shan. Analysis of shopping behavior based on surveillance system. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 2512–2519. IEEE, 2010.

[40] M. Quaritsch, K. Kruggl, D. Wischounig-Strucl, S. Bhattacharya, M. Shah, and B. Rinner. Networked uavs as aerial sensor network for disaster management applications. *e & i Elektrotechnik und Informationstechnik*, 127(3):56–63, 2010.

[41] M. Quaritsch, E. Stojanovski, C. Bettstetter, G. Friedrich, H. Hellwagner, B. Rinner, M. Hofbaur, and M. Shah. Collaborative microdrones: applications and research challenges. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, page 38. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[42] ROTOS. http://www.robcib.etsii.upm.es/index.php. 2016.

[43] R. Rysdyk. Unmanned aerial vehicle path following for target observation in wind. *Journal of guidance, control, and dynamics*, 29(5):1092–1100, 2006.

[44] J. Sgall. Solution of david gale's lion and man problem. *Theoretical Computer Science*, 259(1):663–670, 2001.

[45] T. C. Shermer. Recent results in art galleries (geometry). *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[46] E. A. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. Protect: An application of computational game theory for the security of the ports of the united states. In *AAAI*, 2012.

[47] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on computing*, 21(5):863–888, 1992.

[48] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[49] Z. Tang and U. Ozguner. Motion planning for multitarget surveillance with mobile sensor agents. *IEEE Transactions on Robotics*, 21(5):898–908, 2005.

[50] J. Tsai, C. Kiekintveld, F. Ordonez, M. Tambe, and S. Rathi. Iris-a tool for strategic security allocation in transportation networks. 2009.

[51] UAV. https://www.ndtv.com/india-news/how-drones-whatsapp-were-used-to-keep-the-peace-for-j-k-election-753252. 2016.

[52] J. Urrutia et al. Art gallery and illumination problems. *Handbook of computational geometry*, 1(1):973–1027, 2000.

[53] B. B. Werger and M. J. Matarić. Broadcast of local eligibility for multi-target observation. In *Distributed autonomous robotic systems 4*, pages 347–356. Springer, 2000.