The University Of Sheffield.

Simultaneous Search and Monitoring by Unmanned Aerial Vehicles

By:

Haoyu Zhang

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

The University of Sheffield
Faculty of Engineering
Department of Automatic Control and System Engineering

March 16th, 2019

The University Of Sheffield.

Simultaneous Search and Monitoring by Unmanned Aerial Vehicles

By:

Haoyu Zhang

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

The University of Sheffield
Faculty of Engineering
Department of Automatic Control and System Engineering

March 16th, 2019

Declaration of Authorship

I, Haoyu Zhang, declare that this thesis titled 'Simultaneous Search and Monitoring by Unmanned Aerial Vehicles', and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Haoyu Zhang

Date: 16/03/2019

To a city that changed me a lot

Acknowledgements

First I would like to thank my supervisor Professor Sandor M. Veres, for giving me professional advices and feedbacks. His inspiring suggestions helped me to go deeper into my research. He also gave me full support on the lab resources, which allowed me to do research with full autonomy.

I also want to thank my previous supervior Dr. Andreas Kolling. He was very enthusiastic in giving me guidance and having discussions with me in the early stage of my PhD. He also gave me many ingenious feedback on my work to improve the quality.

I am also grateful for Dr. Jonathan M. Aitken, Dr. Owen McAree, and Mr. Michael Port, who gave me technical help with patience. I also thank the other colleagues who gave me suggestions and support.

I would also like to thank my girlfriend Miss. Lucinda Alice Anne Clegg, who accompanied me throughout my whole writing-up period. Without your love and support this would have been much more difficult. I also want to thank her parents, Mr. Grahan Clegg and Mrs. Sandra Clegg, for treating me with love.

Last but not least, I would like to thank my parents, who always loved me unconditionally. You taught me to work hard to chase my dream, and here I come.

UNIVERSITY OF SHEFFIELD

Abstract

Faculty of Engineering
Department of Automatic Control and Systems Engineering

Doctor of Philosophy

Simultaneous Search and Monitoring by Unmanned Aerial Vehicles

by Haoyu Zhang

Although robot search and monitoring are two problems which are normally addressed separately, this work conceives the idea that search and monitoring are both required in realistic applications. A problem of simultaneous search and monitoring (SSM) is studied, which innovatively combines two problems in a synergistic perspective. The single pursuer SSM of randomly moving or evasive targets are studied first, and are extended to the cases with multiple pursuers. The precise mathematical frameworks for this work are POMDP, POSG and Dec-POMDP. They are all intractable and non-scalable. Different approaches are taken in each scenario, to reduce computation cost and achieve online and distributed planning, without significantly undermining the performance. For the single pursuer SSM of randomly moving targets, a novel policy reconstruction method is combined with a heuristic branching rule, to generate a heuristic reactive policy. For the single pursuer SSM of evasive targets, an assumption is made and justified, which simplifies the search evasion game to a dynamic guaranteed search problem. For the multiple-pursuer SSM of randomly moving targets, the partial open-loop feedback control method is originally applied to achieve the cooperation implicitly. For the multiple-pursuer SSM of evasive targets, the assumption made in the single pursuer case also simplifies the cooperative search evasion game to a cooperative dynamic guaranteed search problem. In moderate scenarios, the proposed methods show better performance than baseline methods, and can have practical computation efficiency. The extreme scenarios when SSM does not work are also studied.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Unmanned aerial vehicles (UAVs) are of natural potential to be utilized in a system to measure information on ground or sea surface. The application of such system can be search and rescue at sea and over land, wild animal monitoring, forest fire monitoring, etc.. For several reasons, it is not always advisable to monitor an area of interest by having a single agent to cover the whole space. Firstly, the sensing ability of each agent is limited by the field of view, sensing range, and resolution of its sensor. Given a required measuring accuracy, the cost of sensor grows exponentially with the required size of footprint. Secondly, the loss of the UAV by failure or hostile attack can be devastating to the whole mission, if there is only one UAV. Thirdly, a single sensor is susceptible to occlusions caused by the terrain, buildings, or the curve of the earth.

There are two main ways to address these issues. One way is to make use of the mobility of the UAV agent. An agent with a small sensor footprint can move around and explore the whole environment, thus covering each part intermittently. This reduces the requirement on the size of sensors footprint, and can be more flexible to focus sensing on areas of higher interest. Another way is to divide the whole space into partitions, and cover them with a set of agents with cheap sensors. Such system should be of inherent robustness against losses of individual agents, thus improving its reliability in an adverse environment. Having multiple scattered sensors can also overcome the influence of occlusion.

In most cases, the two concepts can be combined, to have a system of multiple moving sensors, to cover and measure the environment. To exploit the potential of such a system, some problems remain to be solved. Firstly, in a dynamic environment with spatial and temporal information, it needs to be decided about what trajectory the agent should follow to schedule the sensing resources, and to obtain and update the real time information of the whole environment. Secondly, with multiple agents working for

a shared mission at the same time, given limited communication, the agents need to cooperate to optimize the overall performance of measurement.

1.1 Simultaneous Search and Monitoring of Mobile Targets

In this section, some more specific scenarios are focused upon. In a situation where a ship capsized in the sea and the sailors are drifting on the water with life vests. To achieve a fast-response rescue, we need an UAV system to reach the incident area to search and locate the sailors, and report to the rescue team. This is illustrated in Figure 1.1. In a natural reserve, we want a system to study the behaviour pattern of several endangered animals.

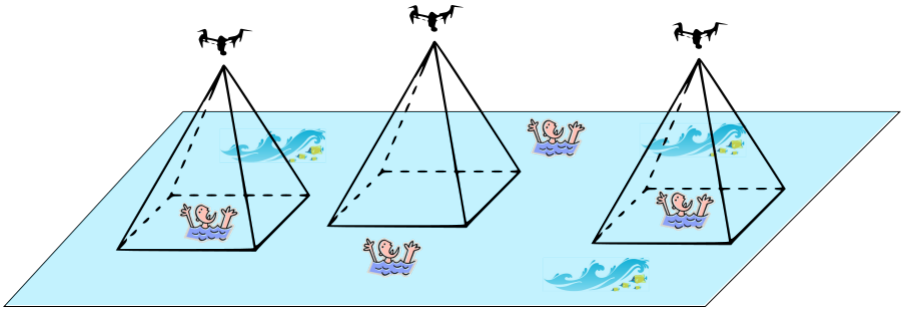


FIGURE 1.1: application of an UAV system in sea rescue

In these applications, we want to obtain and maintain an up-to-date knowledge about the ground and sea targets. The target detection sensors are of limited resolution, and the whole environment to observe is normally large. So it is impossible to cover the whole area by a static sensor. Instead, multiple mobile sensing agents are needed.

Numerous prior work has studied this problem area. They are mainly divided into two categories: one is searching for unknown targets to obtain their location, the other is monitoring known scattered targets to update their information. In both categories, the problem formulations are further divided by how fast the agent can outrun the targets, how many targets each agent needs to cover, and how big area of the environment can be measured at the same time. In search missions, the searcher may build a fixed formation to cover the whole area statically, or sweep in a fixed pattern, or explore dynamically, to achieve fastest or best chance of detection. In a monitoring missions, the pursuers may track one individual target, or cover multiple ones, or traverse them in a sequence, to maintain the knowledge about them. search and monitoring by UAV system are shown in Figure 1.2 and 1.3.

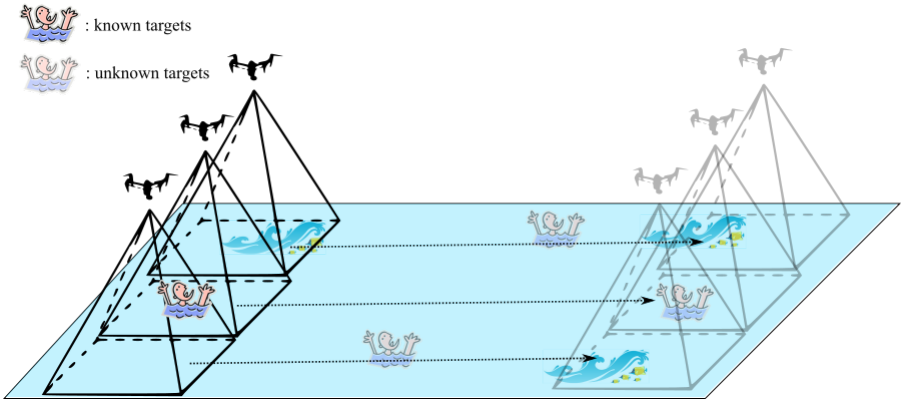


FIGURE 1.2: search in sea rescue

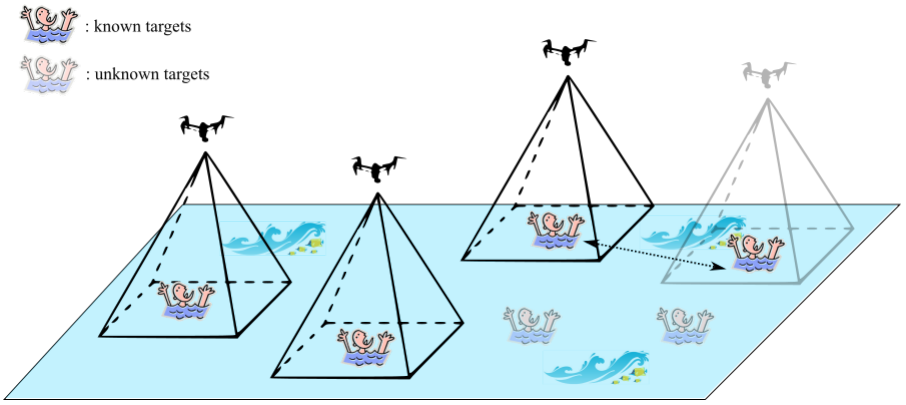


FIGURE 1.3: monitoring in sea rescue

In the scenarios mentioned at the beginning, however, such dichotomy does not meet the actual demand. For example, if we send the UAV fleet to search for the lost sailors, after their location being detected and reported, they may be pushed away by sea waves, and when the rescue team arrive they may not be found. If we send UAV fleet to monitor and update the location of the sailors who are currently known, the unfound targets may not be detected and rescued. This applies to other kinds of targets as well. Thus we can imply that the search and monitoring are both required. Unknown targets should be found and known ones should be kept under surveillance.

There do exist some works which attempted to tackle this problem [1–5]. In [1], the search and tracking (SaT) of one single target is studied. However, with only one target being searched and monitored, the search and monitoring are in an interleaving pattern rather than simultaneous. Thus the planning and execution of search and monitoring are separated into two independent tasks, and do not need to be combined. Therefore this work can not be applied in the scenarios with multiple targets. In [3–5], target search and monitoring of multiple targets are combined as a multi-task planning, which

is solved as a task assignment problem. In such a way, it is a trade-off between tasks, which do not exploit the synergy of search and monitoring. In [2], search and monitoring of multiple targets are achieved in a single mission, but the agents only try to reduce the overall area of the worst case distribution of all targets. Therefore, in the planning, search and monitoring of any individual target is not specifically considered, which also ignores the potential cooperation between search and monitoring.

To address this problem, this thesis studies a simultaneous search and monitoring (SSM) problem, in which a single or multiple UAV searchers are required to continuously search and monitor several mobile targets in a large environment. The pursuers try to update the location information of as many targets as possible, through searching for unknown targets while monitoring known ones in parallel. The concept of SSM is shown in Figure 1.4, and will be further discussed in section 3.4.

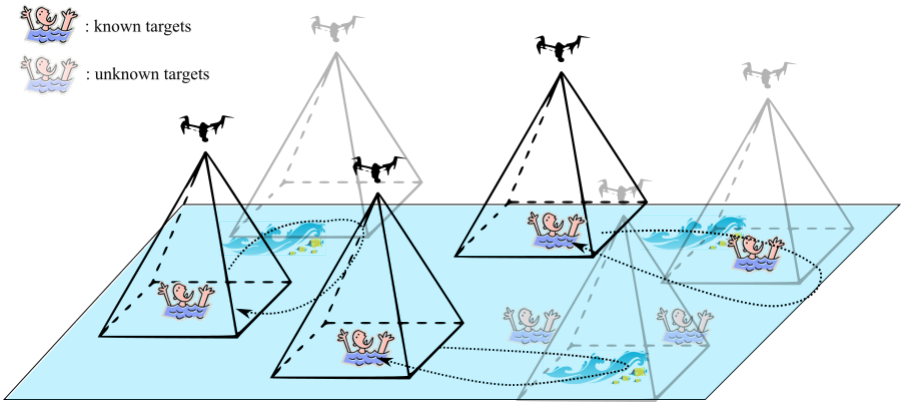


FIGURE 1.4: SSM in sea rescue

1.2 Problem Statement

The detailed problem to solve is discussed in this section. As introduced in Section 1, to achieve simultaneous search and monitoring with multiple UAV agents, it requires sensor scheduling and strategy planning for each individual pursuer, and cooperation between agents.

1.2.1 Sensor Scheduling and Strategy Planning

In most of the search and surveillance problems, the information to acquire is normally uncertain, such as the probability distribution of the target presence, or the occurrences of stochastic events. Nevertheless, the goal for the agents is to obtain a general knowledge of the information, such as the expected number of detections, expected monitoring or

service levels, overall awareness, or information entropy. Therefore in these problems, individual stochastic incidents do not affect the overall reward. Thus the gain of the mission becomes deterministic and predictable w.r.t. the plan of the pursuer. Then a fixed plan to schedule its sensing resources will be enough to solve the problem.

However, in SSM, the specific information about every target is concerned about. Thus each contingency, such as detecting a new target or losing a known one, may dramatically change the situation in the environment, and so does the expectation of reward. Therefore reactions to these events need to be considered in planning, to take into account the influence of possible future events. When doing the planning, the agent should search through a decision tree, which contains the branchings triggered by possible target behaviours, and find the best reactive solution. It can be seen that branchings will substantially increase the computational complexity in this problem, which is the main difficulty to overcome. In [6], a problem of searching for a single target is tackled with both deterministic method and probabilistic method, and the deterministic method is shown to have better performance than the probabilistic method. This is because, if there is only one target to be searched, the search terminates when the target is found, thus a deterministic planner can simply assume that the target would not be found during search. Hence the deterministic search can be more efficient compared with probabilistic method which is more sensitive to uncertainties on target model. Therefore, this result does not apply to our scenarios when multiple targets need to be simultaneously searched and monitored.

In SSM, the targets may move randomly regardless of the actions of the pursuer. If the probability of each target motion is known, the chance of each branching at each state can be predicted forwardly to the future. This problem can be formulated as a Partially Observable Markov Decision Process (POMDP). In other scenarios, the targets can also proactively evade the pursuer. In such a case, each side of the pursuit and evasion needs to consider the possible strategy of the other side and plans its best policy, which is a Partially Observable Stochastic Game (POSG).

The first problem to solve is, given a type of target behaviour, a single pursuer should plan and execute a strategy for its motion and sensing, thus to obtain and update the best available knowledge of target information. The kind of target behaviour can be either random or evasive.

1.2.2 Multi-agent Cooperation

To exploit the potential of a multi-UAV system, a centralized planning can be an ideal solution. A chief agent can gather and fuse measurements from other agents, do the planning for all the agents, and allocate each sub-plan to the corresponding UAV. Nonetheless, this kind of robot network requires high bandwidth communication, which is not scalable. It is not reliable and robust in an adverse environment. Some prior works do decentralized cooperation with heuristic approaches, either by partitioning the environment and assign to pursuers, or by having a myopic guidance law for each agent. The heuristic methods can be scalable and easy to implement, but are difficult to provide theoretical guarantee of the performance.

Thus a decentralized and non-hierarchical multi-robot system is expected, which only requires limited communication. The planning should have look-ahead ability. In such system, each agent do the planning in a distributed way, taking into account the information acquired through its own sensing and the communication with other agents. Such distributed planning can coordinate the scheduling of every agent, thus can achieve a synergy and avoid redundant overlap of efforts. This problem is formulated into a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), or a Partially Observable Stochastic Game (POSG), for the cases with randomly moving or evasive targets. This thesis uses game theoretic methods to plan the distributed cooperation.

1.3 Contributions

To solve the problems mentioned in Section 1.2, this thesis presents the following contributions:

1. **Combining search and monitoring as a cooperation**. Instead of having an intuitive combination by treating two tasks separately and having a trade-off between them, this work combines the search and monitoring as a cooperation. In a dynamic situation, a target can change between know and unknown by being detected or being lost. Thus the two problems are interconnected, and should have a synergy to better react to possible detection or loss of a target. This combination is achieved by building an united goal. This is the first work which combines the search and monitoring in an explicit and synergistic way.

2. **Solving POMDP with a heuristic reactive policy**. To have an online solution of POMDP, a novel policy reconstruction method is proposed. It makes room for decomposing a policy and having intuitive approximation. A heuristic branching

rule is proposed to approximate the sensible reactions that an optimal strategy would make. It is combined with policy reconstruction to generate a heuristic reactive policy. Such policy shows better performance than two other baseline methods, and have practical computational efficiency.

3. **Solving the search evasion game as a dynamic guaranteed search problem**. It is intractable to solve search evasion game precisely, so an assumption about the information available to the targets is made. This simplifies the search evasion game to a novel dynamic guaranteed search problem. Such dynamic guaranteed search works better than a conventional fixed pattern guaranteed search.

4. **Solving the decentralized cooperative SSM with partial open-loop feedback control** The concept of partial open-loop feedback control is innovatively applied on the distributed cooperation. It allows the agent to focus on local information, but still achieve the cooperation implicitly. The local policy can then be designed with heuristics as in the case of single pursuer SSM. The advantage of such cooperation is achieved in simulation.

1.4 Publications

The research described within this thesis represents the original efforts of the author. Some of it has previously been published in the form of peer-reviewed papers. These papers are as follows.

1. **H. Zhang**, S. Veres, A. Kolling. "Simultaneous Search and Monitoring by Unmanned Aerial Vehicles", in *proceedings of 2017 56th IEEE Conference on Decision and Control,* IEEE, 2017, pp. 903–910

2. **H. Zhang**, S. Veres. "Simultaneous Search and Monitoring of Evasive Targets by an Unmanned Aerial Vehicle", in *proceedings of 2018 12th International UKACC Conference on Control,* UKACC, 2018, pp. 277–282

Paper 1 consists of the content from Chapter 3 and 4. The content of Chapter 5 comes from paper 2. The combination of Chapter 4 and 6 is in a paper which is about to be submitted to a Journal.

1.5 Outline

The thesis is organised as follows.

1. The Literature Review of related work is done in Chapter 2. It introduces a general background on robot search and pursuit evasion. Such background demonstrates the separation of search and monitoring, from which this thesis is inspired. It then reviewed the problems of POMDP, POSG, and Dec-POMDP, which are the mathematical frameworks of this work;

2. The problem formulation of this research is represented in Chapter 3. The models of the environment, the pursuers, and the targets are build first. The concept of simultaneous search and monitoring is then presented;

3. The simultaneous search and monitoring between a single pursuer and multiple randomly moving targets is studied in Chapter 4;

4. The simultaneous search and monitoring between a single pursuer and multiple evasively moving targets is studied in Chapter 5;

5. The Cooperative simultaneous search and monitoring by multiple pursuers is studied in Chapter 6, which extends the results from Chapter 4 and 5.

6. The conclusion of this thesis is drawn in Chapter 7.

Chapter 2

Related Studies

Robot search and pursuit evasion is a problem for a single or multiple robot system trying to detect or capture one or more targets [7]. There is one category of pursuit evasion game, that both the pursuers and the evaders have unlimited sensing range and are of comparable speed, such as tiger and lady [8], cop and robber [9], and homicidal chauffeur game [10]. The goal of the pursuers are to catch the evaders and vice versa for the evaders, and the problem is normally solved with differential or combinatorial methods. These games assume perfect information for both sides, and the planning focuses on the kinetics of the agents.

However, for the problem in this thesis, it focuses more on the information gathering in a unknown or adversarial environment, where the pursuers have limited sensing range or imperfect sensor. This review does not include the pursuit evasion game with perfect information, but focuses on the problems which are more relevant to this thesis. Section 2.1 first introduces the general background of robot search and pursuit evasion, by reviewing the related works in a task-oriented perspective, which includes target search, target monitoring, and multi-robot cooperation. The motivation of this work, which is to combine search and monitoring, is inspired by such background. Then, in a methodology-oriented perspective, Section 2.2 discuss some underlying problem framework which are commonly involved in these scenarios, including Partially Observable Markov Decision Process (POMDP), Partially Observable Stochastic game (POSG), and Decentralized Partially Observable Markov Decision Process (Dec-POMDP). These problem framework and the solutions will be the foundation of the methodology in this thesis.

2.1 Scenarios in Robot Search and Pursuit Evasion

2.1.1 Target Search

When some targets are scattered and hidden in the environment, and when there is very few information known a priori, the whole environment can be treated in a general way. A distribution of information can be used to estimate the possible target location, rather than using a specific position. Given such distribution, an UAV system can be deployed into the environment to detect the possible hidden targets. The targets may be static or mobile. For mobile targets, they can move randomly or evasively. Based on the sensing ability of the pursuer and the nature of the target behaviour, there are six main categories of problem formulations for target search: static coverage search, dynamic coverage search, probabilistic search, search evasion game, guaranteed search, and awareness coverage.

2.1.1.1 Searching Static Targets

When the targets are statically scattered in the environment, the problem is relatively simple. The pursuer agents just need to cover the environment constantly or intermittently with sensor, thus putting all targets under measurement.

Static Coverage

When there are enough number of agents, so that the union of the sensor footprints can cover the whole area of interest, then a static distribution of UAVs is sufficient for the search mission. It is very obvious that if the sensing of all agents is perfect, all the targets will be found right after the area is fully covered. Thus the works on this study mainly focus on the problem with the assumption of imperfect sensing. In these works, the targets may be detected at each time instant with a probability, and such probability is related to the set-up of the UAV formation.

A common assumption made in their work is that, at certain position, the probability of detecting a target increases with the estimated density of targets and decays with its distance to the closest sensor. A strategy is studied by many works which deploys the agents to a set of static positions, to achieve the coverage with highest expectation for target detection. In [11], the author built an utility function to represent the gain for total covering effect, with respect to the location distribution of agents. The utility function can be decentralized to each agent, then the gradient of the local utility function can be the navigation law for each agent. Such navigation law drives the agents toward a set of positions, which is a local optimal solution. The author made an assumption that

the cost function of an agent to sense certain point increases proportional to the square of the distance between the agent and that point. Then the author obtained the result that, the global optimal distribution for agents is the set of centroids of all voronoi cells. It is a very promising result because the optimal deployment, which is simply the centroid positions, is easy to compute in a distributed manner. The navigation law for each agent can be a distributed gradient function. Thus the complicated real time optimization problem is avoided. In [12], the author extended the work in [11] by introducing adaptation for the information density of targets. In this work, the information density for target, which can be viewed as the probability distribution for target existence, is assumed to be unknown, but the density at certain point can be measured by the agent nearby. The Author designed a local adaptation law for the information density function for each robot, and a parameter consensus law is devised to obtain a global estimation. Then a control law can be designed to drive the agents towards the estimated centroid of its voronoi cell, and such estimation is keep being updated by sensing. It is guaranteed that with such control and estimation law, the agent distribution will converge to the set of global optimal locations. Nevertheless, the results of above works are based on the assumption that the cost function of an agent to sense certain point follows a specific pattern, which may not necessarily be accurate in all applications. Thus the application of such an elegant result is limited.

The static coverage problem is further discussed in works [13–16].

Dynamic Coverage

If the area of interest is large, or if there are not enough UAVs available, the agents cannot sense the whole area with a static formation. Thus the agents should explore the environment dynamically to find out the hidden targets. When the targets are static, they can not enter the cleared area to cause recontamination. If the sensor of the agent is perfect, any target within its sensor footprint can be detected immediately. Then the agents can find out all the targets by clearing the area for just once. The goal of path planning should be to have a set of paths to sweep the environment in the shortest period of time. This problem can be solved with geometrical methods. In [17], the author designed the path planning policy for single or multiple robots to visit each point in the area without overlap and revisiting any point. However, the starting point for the path of each agent should be calculated by the planning, rather than having an arbitrary initial set-up. This limits its flexibility. In [18], the author proposed a computational feasible algorithm, which constructs a set of trajectories for agents to cover the whole area, given an initial position of each agent. These paths do not overlap and are of equal length, thus the author proved that it can make a critical contribution in minimizing the total time for coverage. In [19], the author designed the

exploring policy with contracting polygons, thus covering a polygon area with a single UAV without overlapping of its path.

2.1.1.2 Searching Mobile Targets

When the targets are moving in the environment, a target may go back to a previously searched area to cause recontamination. In such scenario, a dynamic search is needed, in which the pursuit strategy should take into account possible target behaviours during the search.

Probabilistic Search

In some cases, the targets may move independently from the pursuer, such as some drifting sailors waiting to be rescued, or a migrating animal flock to be found. For this kind of targets, the random motion model is usually applied to represent their dynamics. In such model, at each time step, the target may move between neighbouring cells with certain probability. Then the estimation of target presence can be constructed as a probability distribution. Given an initial estimation of the distribution and a sensing model, a Bayesian model can be constructed for the update of target probability distribution, given the sensing history of the pursuer [20–22].

After constructing the target model, the evolution of the estimation can be predicted w.r.t the planned motion of the pursuers, in an one-sided manner. The likelihood of a detection at each time step can be predicted as well. Thus under this set-up, the objective of target searching is to optimize the probability of finding the targets or to minimize the time before detection. In some other works [23, 24], the probability distribution is transformed to be the information entropy distribution, to describe the uncertainty of target estimation. The goal of search is to reduce the overall uncertainty level in the environment.

However, it has been proved in [25] that, in probabilistic search, maximizing the probability of target detection is NP-complete, and the problem to minimize the time for detection is NP-hard. As a result of the intractability of probabilistic search problems, a lot of researches focus on finding near-optimal solutions to the variations of the above two optimization problem [7]. The myopic method, such as heuristic guidance law, is a common approach circumventing the time-consuming path planning. It steers the robot to a best immediate direction, to try to achieve a good overall performance over time [20, 23, 24, 26]. In other works, the robots do a path planning over a time horizon, while introducing some pruning or simplification to reduce computational complexity [27–29].

In [24], the author built the information entropy map and the update rule to estimate the target presence. Then a gradient based navigation law is designed to steer the agents to the direction with largest gradient of information entropy, trying to reduce the total uncertainty of the estimation of target presence. In [23], the same objective is applied, which is to minimize the total entropy of the agents regarding target location estimates, in an application of multi-target tracking and surveillance. In [20], the authors study the case of target search for multiple heterogeneous agent and multiple randomly moving evaders. The authors propose an heuristic pursuit policy, which drives the agents to the position with largest probability of target presence. The authors prove that this policy can guarantee at least non-zero probability of capturing the targets within certain period of time, and equal to one probability of capturing within finite time. The authors validate its result by simulation and experiment. However, this guarantee of performance is trivial, because as long as the targets move randomly, any pursuit policy can guarantee a non-zero likelihood of detection over a certain period of time, for the fact that a target always have a probability to move into the sensor footprint of the pursuer. This is a problem shared among almost all myopic search approaches, that it is very difficult to provide a significant theoretical guarantee for its performance. However, the flexibility and scalability to different situations make it still a practical and popular method.

The non-myopic method, such as path planning, can predict and optimize the expectation of detection in a more rigorous way. However, the computational difficulty is the main limit of its application. The branch and Bound approach is commonly applied to do path planning with reduced complexity. In [27–29], the path is generated by setting the current position of agent as the root, then enumerating the search space to expand the search tree. The branches would be pruned if have their upper bound of estimated reward to be lower than the lower bound of current best branch. Iterating this step until it reaches the end of the search horizon, a policy tree can be constructed. Although the pruning can reduce the computational complexity, this method is still computationally inefficient and its computation time may be subject to the size of search space.

Other works about probabilistic search are in [3, 30, 31].

Search Evasion Game

For some adversarial targets, such as some criminals to catch, they may pro-actively evade from the pursuer to avoid detection. The evasive behaviour makes it a game playing problem. In this case, the evolution of the game can not be predicted w.r.t the actions of the pursuer, in a forward induction manner. It is because this is a two sided planning problem, in which both the targets and the pursuer plan their own actions according to their observation of each other, and both need to search through the whole decision tree to the end of time horizon, to evaluate each action to take. Thus this

coupling makes the pursuit policy much more difficult to plan compared with one sided planning.

Considering the fact that an aerial agent can only sense part of the environment, and that it is a two-sided game playing, the search and evasion game between aerial pursuers and ground targets can be modelled as a Partially Observable Stochastic Game (POSG). The exact solution of POSG is very difficult [32]. Thus, given that search evasion problem normally has a big state space, there is few work to solve this problem in the precise form of POSG. Therefore, some methods were applied to avoid the enumeration in backward induction, such as sampling [33] or myopic approach [30]. In [33], a set of samples of future states are taken from the search tree, and by evaluating the rewards in these sampling states, the reward of a plan can be approximated. Nonetheless, the accuracy of such estimation depends on the number of samples. In search evasion game where the search tree expands exponentially with time horizon, such method is still computationally difficult. In [30], the search evasion game was studied as a POSG, but just for a single time step. The Nash Equilibrium was taken as the solution. The same with other myopic methods, such approach can not guarantee a long term performance.

For the above reason, the most common simplification method for search evasion game is to approximate the evader policy by a heuristic target model, thus disentangling the evader planning from the planning of pursuit policy. Most works assume a pattern that the targets will follows to hide away from the pursuer. In most cases, it is assumed easier for the ground targets to sense the aerial pursuers, thus the targets can react and elude, from outside of the sensor footprints of the pursuers. In some works [34–36], the targets are driven by the synergy of potential forces from the pursuers and obstacles. In [37, 38], the evaders move like a Reactive rabbit, which dodge from the pursuers only when they are close enough. The agents can plan their strategies with the assumptions on the target model, which is an one-sided planning.

Guaranteed Search

Both probabilistic search and search evasion game try to achieve an efficient performance, and demand intense computation. Hence an alternative method is to obtain a worst case performance. A worst case assumption about the behaviour of evaders can be made, and thus generating an easier solution, which can guarantee a certain level of performance. The idea of recontamination is normally applied, which assume the unknown targets can move backed to the previously cleared area with a certain speed. Thus the agent should either block the possible routes of recontamination, or clear the re-contaminated area, thus guaranteeing that the targets will be found in any case.

For the indoor search and pursuit evasion operation, the space usually consists of individual rooms and the connecting corridors, and the agents in each room can usually sense the whole room if without obstacles. Then, the environment can be modelled as a graph. For the search and pursuit evasion on a graph, the target is assumed to move with unbounded speed [39, 40], or with a speed bound [41]. The goal of study is to design the pursuit strategy to clear the graph, given a graph and multiple players [41], or to determine the minimum number of pursuers and the associated strategies which are required to guarantee detection [39, 40].

For UAVs, most of the operation environment will be an open outdoor area, which is continuous and can not be modelled as a graph. In such case, the targets can be assumed to move with a bounded speed. The worst case assumption is that the cleared area will be re-contaminated with such speed bound. The agent can move in a pattern which will clear new areas and stop the propagation of recontamination at the same time. Such pattern of trajectory can be back-and-forth lines in parallel [42, 43], or a spiral line [44], or taking advantage of the shape of the environment [45]. By having an overlap of the covered area between each round of sweeping, the agent can keep moving further in the environment, and the recontamination can be cleared before it reaches the clean zone. In such a way, the whole environment will be searched without recontamination, and the hidden targets can be detected with certainty.

It can be seen that such worst case assumption can largely reduce the computational complexity, and assure performance to some extent. But it can be too conservative when efficiency is preferred.

Awareness Coverage

Another alternative to circumvent complex computation is by focusing on improving and maintaining the awareness level of the whole environment, instead of specific targets. In the method of awareness coverage, an awareness model is built to estimation the uncertainty of possible target existence at certain location. The awareness level accumulate when certain region is under measurement, or decay otherwise. In work [46] and [47], some exploring paths are designed for agents to sweep the whole region thus to maintain the overall awareness within a satisfactory level. In this kind of works, the concept is to make sure every piece of area will be measured after a certain time interval, thus to reduce the uncertainty of its knowledge for the whole area. The awareness model is deterministic w.r.t the search effort, so the path planing is relatively simple. But it is an indirect approach in terms of its effort to detect targets, because reducing uncertainty does not necessarily equal to increasing the chance of target detection.

2.1.2 Target Monitoring

When the UAV pursuers have already known the location of some moving targets with certain level of belief, they may try to update the location of these targets to keep the information valid. Different scenarios of target surveillance depend on the sensing range of the pursuer and the density of the targets. These scenarios can be: target tracking, target covering, and target visiting.

2.1.2.1 Target Tracking

When there is only one target to measure, the UAVs should keep updating the target's position, by keeping the target within sensor vision. In work [48], a team of fixed wing UAVs try to track a ground target with a fixed moving speed. The lower speed limit of the UAVs is faster than the speed of the target. Thus to keep the target within sensor footprint, the UAVs should circle around the target. The author proposed four guidance law for a double UAV team to track a moving target, ensuring that the UAVs keep a constant distance from the target and maintain constant angular separation. However, its guidance law is based on the knowledge of the constant moving speed of the target rather than real time measurement. The tracking of a manoeuvring target based on measurement is studied in [49]. In the case when the agents are able to be relatively static to the target, the circling pattern still has its advantage. In [50], one single static or dynamic target is tracked and sensed by a team of agents, of which the objective is to reduce the uncertainty of the target information. The objective function is built as the inverse of the covariance of the sensing error. The authors found out that the optimal placement for the agent team is a set of equally spaced angular positions around the target. Then the author designed an cooperative formation control law to steer the agents to the optimal placements with proven convergence. The author proved its superiority to the static placement.

2.1.2.2 Target Covering

When there are more targets than pursuers, each agent should not just focus on a single target. In the scenario of target covering, the target is crowded, and the UAV pursuers try to keep as many targets within its sensor footprint as possible. In [51], the authors addressed the CMOMMT problem (Cooperative Multi-robot Observation of Multiple Moving Targets). The primary focus is on developing the distributed control strategy for the agents, given the locations of nearby robots and targets. The strategy should allow the team to minimize the total time in which targets escape observation. The robot

is assumed to move faster than the target, and the targets are densely scattered. Each robot can sense a relatively big area. Thus the robot should focus on currently covered targets rather than exploring to find new targets. A heuristic method is proposed, which is similar to the potential field method. It takes into account the attraction of targets and repelling of the fellow robots, to form the weighted total force to drive the movement of the robot. The simulation and experiment showed the superiority of this cooperative algorithm to random move or local cooperation. In [52], the author extended the work in [51], and a Behavioral CMOMMT algorithm were proposed, to overcome the possible situation in A-CMOMMT which one robot follow two targets that move in opposite direction, eventually losing both. Three modes for the robots were designed, which are: follow, help, and explore modes. Different kinds of prediction algorithm were used to predict the lost time for the target. Thus for the exploring robots, once the difference between the time to capture the target and the time to loss exceeds a defined threshold, the robot stops exploring and starts to move to put the targets into the area covered by its sensor. The simulation and experimental result shows significant improvement compared with A-CMOMMT. However, for the above two works, the local heuristic force could not guarantee the global performance and may get trapped by local minima.

In [53], the agents can have unlimited view and imperfect sensing with gaussian noise. The authors modelled the sensing and updating process based on Kalman Filter. The objective function was expressed in the form of total uncertainty, and some paths are designed to deploy the agents to certain positions to optimize objective function.

2.1.2.3 Target Visiting

In some scenarios, the targets are scattered sparsely in the field, thus the agents can not cover enough targets at a time. Then it is necessary for each agents to leave the targets it is currently covering, to try to re-detect other targets to update the information of them. Some works choose to formulate the problem as a Dubins Travelling Salesperson Problem (DTSP), which is to find a shortest path for the UAV to visit each selected targets in a chain, to minimize the escape probability of each target. If the agent can go back to each target within certain period of time, the target would not escape from the sensor footprint, given the assumption that the targets move with limited speed. In [54], the author assumed that there is an upper bound for the target speed, and designed an optimization law for the DTSP, so that the UAV can traverse each target in every minimal time period to ensure that each target always stays within the footprint when the UAV comes back. In [55–58], the DTSP problem is discussed in different ways.

If achieving an efficiency of measurement is preferred than guaranteed success in re-visiting, the fact should be considered that a target may get lost if the agent fails to re-detect, which may invalidate an old plan. Thus the reaction to such loss should be considered in planning, to focus on measuring the rest of the targets. The [59] and [60] studied the Partially Observable Markov Decision Process (POMDP) for such a problem. The target model is built based on belief state, and the sensing model is based on Kalman Filter. The agent optimizes its trajectory using Nominal Belief-State Optimization (NBO), to traverse each target, thus to stop the uncertainty to grow and achieve best belief of the target location. However, this two works assumed that the targets move with known average speed or average acceleration, which may not be practical. In [61], a similar problem was studied, except that the motion of the targets and the pursuer are constrained on a road map. The authors then designed some Marco-Actions, which are possible series of actions taken by agent, to simplify the action space of the pursuer. The best Marco-Action is planned by a tree search.

2.1.3 Multi-Robot Cooperation

In the above works introduced, there may be more than one pursuers in the mission. With proper coordination, multiple agents may be able to have synergy over the same task, which should outperform simply adding up their work. Plus, if there is an unnecessary overlap among the effort of the pursuers, the redundant resources are wasted. Hence, the cooperation among the agents should be designed to exploit the advantage of a multiagent system and avoid redundancy.

2.1.3.1 Cooperation with Myopic Methods

In the works which apply myopic methods for the motion control of the agents, the current information of neighbouring agents can be incorporated into the local control law, thus considering the cooperation in a heuristic way. In some works, such as [11, 12, 50, 52], the cooperation laws are carefully designed, which can guarantee the global performance. Nevertheless, in most cases, such theoretical guarantees are based on specific problem formulations, which are not easy to be applied in general missions. Thus in other works, the global performance is not considered. In [51], a cooperative law based on potential force is designed, which is intuitive, but the author did not justify it with theoretical proof. In some works such as [24] and [20], the local control laws are designed based on greedy search, but the cooperation is not considered. The cooperation is achieved implicitly by having agents scattered initially in different locations, thus resulting in non-overlapping trajectories.

The myopic methods for cooperation normally do not require large amount of computation, thus is easy to implement. Plus it is normally a local control law, thus is inherently distributed, which makes it scalable and robust. But the difficulty of having theoretical proof of performance limits its application.

2.1.3.2 Cooperation with Task Allocations

Another approach for the coordination of the pursuers is task allocation. In this method, a hierarchical task formulation is built. Different targets, or different parts of the environment, are divided. The cost and reward of possible actions, which may be taken on these targets, are calculated in a heuristic way. Thus each possible plan on each target is built as a task. The pursuers then find the most efficient way of allocating tasks to available agents, based on the estimated cost and gain of tasks. The agents then design the detailed local control laws to accomplish the assigned task.

In [3, 29, 62], the task allocation is achieved in a centralized way. A central agent in the team sets up the plans based on global information, and then distributes the local task to other following agent. This requires high communication bandwidth and reliability. In [23] and [54], the task allocation is decentralized, in which neighbouring agents coordinate tasks based on local planing and communication. In some search problems, the task allocation can be simplified to be the partitioning of the area, and letting each agent to search in one partition [63, 64].

Task allocation method is intuitive and not complicated, but it relies on abstraction of different tasks, thus is difficult to accurately model and evaluate each plan.

2.1.3.3 Cooperation with Non-Myopic Planning

To avoid the local minima problem of myopic methods, and to do planning more precisely with mathematical rigour, approaches with look ahead ability are widely studied for the multi-agent cooperation. A Non-myopic Strategy is much more complicated than an abstract task, thus it would be a heavy burden for the communication, if there is a central agent to plan all strategies online and do the assignment in real time. Therefore in most works, they either have an offline planner to do the centralized planning of distributed plans, or do a distributed online planning with limited or no communication.

In [65, 66], a centralized Multiagent POMDP (MPOMDP) was studied for a multi-robot information gathering problem. The full and perfect communication between the agents is assumed. The control of the team can be viewed as centralized. A joint strategy was planned offline, and the robots execute such policy, with their information always

shared in real time. In [67], a multiagent target search problem is studied. The perfect communication is also assumed. Except for a centralized planner similar to [65, 66], the authors also designed an implicit coordination, in which each agent shares its current plan during execution and re-plan its own strategy considering the received plans from the other. This implicit coordination works better than the case without cooperation, and are more scalable than the centralized planner.

The full and perfect communication may not be realistic in some scenarios, such as indoor environment where the walls may block signals, or in a large outdoor environment where there is a range limit of communication. The multiagent cooperation without or with limited communication can be modelled as a Partially Observable Stochastic Game (POSG) or a Decentralized Partially Observable Markov Decision Process (Dec-POMDP). In [68], a Dec-POMDP is studied for the multi-robot search and pursuit evasion. With no information sharing, each agent has to estimate the observations and policies of the other agents, and incorporate such estimation into its own policy. The authors took advantage of the fact that the mission is symmetric to each agent, thus the strategy should be the same for every robot. Such common strategy was planned offline using a heuristic policy improvement. In [69], a POSG is also studied for a cooperative search problem. The policies are designed online, which are simplified to be a set of Bayesian games. Solving POSG or Dec-POMDP can be difficult [32], thus for the multiagent cooperation with non-myopic planning, simplifying the problem to allow practical application is an important part of the study.

2.2 Methodologies

The above section reviews some relate works in the broad background of robot search and pursuit evasion. Although the set-ups of these works are different, we can see that there are some common difficulties to be addressed. For the pursuer to search or monitor the targets efficiently or assuredly, because of the unknown and uncertain locations and actions of the targets, the pursuit strategy should include the reactions to the stochastic or adversarial behaviours of the targets. Also, when there are multiple pursuers available, the coordination between them should be designed, to have synergistic efforts without redundancy. Therefore, in this section, the Partially Observable Markov Decision Process (POMDP) is introduced as the framework to study the search and monitoring of randomly moving targets. The Partially Observable Stochastic Game (POSG) and the Decentralized Partially Observable Markov Decision Process (DEC-POMDP) will then be discussed, for the search and monitoring of evasively moving targets, and the cooperative search and monitoring.

2.2.1 Partially Observable Markov Decision Process

A scenario is considered, where a pursuer with limited sensing capability tries to search for a single or multiple randomly moving targets. The stochastic target movement make the system state a Markov Chain. The partial observability of the pursuer means that it has to hold an estimation of the system state, which is called the belief state. It is suitable to model this problem as a Partially Observable Markov Decision Process (POMDP) [70]. The POMDP is a tuple $\langle S, A, T, R, \Omega, O \rangle$ [71], where

1. S is a finite set of states of the world;
2. A is a finite set of actions;
3. $T : S \times A \longrightarrow p(S)$ is the state-transition function, mapping from a previous world state and an agent action, to a probability distribution of next world states;
4. $R : S \times A \longrightarrow R$ is the reward function, mapping from a current world state and an agent action to an immediate reward;
5. Ω is a finite set of observation of the pursuer;
6. $O : S \times A \longrightarrow p(\Omega)$ is the observation function, mapping from a current world state and an agent action, to a probability distribution of agent observations.

For a POMDP with finite horizon from t_0 till t_f, the objective value function can be formulated as:

$$V = E\{\sum_{t=t_0}^{t_f} R(s_t, a_t)\} \tag{2.1}$$

which is the expected sum of the rewards within the time horizon, given the possible states s_t and the actions a_t at each time instant.

Let Y_t to be the observation history. Then the solution of a POMDP is a policy $a_t = \pi(Y_t)$, which is a mapping from the history of sensing to an action, so that to maximize the value V. The POMDP can be solved either offline or online, and each is suitable for different scenarios. However, it is proven in [72] that, solving POMDP precisely is PSPACE-hard. Thus most of practical solutions should include simplifications to some extent.

2.2.1.1 Offline Solutions of POMDP

Majority of the works on the POMDP focus on the offline solution, which enumerates the whole state space, and calculate a global strategy which is optimal for all possible belief state. Such a policy can be executed in real time with no re-planning, and can always take the best action at all conditions. Comparing with the fully observable Markov Decision Process, the POMDP has continuous belief state which induce a continuous belief state space. Fortunately, [70] has proved that the optimal reward function of POMDP is a piecewise-linear, convex function of the current state probabilities of the internal Markov Process. This property significantly simplifies the offline solution.

Two common methods of offline solution are value iteration and policy iteration. The value iteration is a backward induction method, which builds up the search tree from the leaf nodes [70]. It enumerates the value of every leaf node, and then propagate the value backwards to upper nodes with Bellman backup operation. A globally optimal policy is build iteratively in such a way. Such process includes considering the whole belief space, thus can be computationally infeasible. Some methods try to simplify the belief space in different ways, such as point-based value iteration [73–75], point-based heuristic search value iteration [76], and belief compression [77].

The policy iteration is another approach, which does not build the search tree step-by-step [78, 79]. Starting with an initial candidate policy, in each round, it adds an incremental modification to the candidate policy, and evaluate its value by doing a Monte-Carlo simulation. The modification is accepted if there is an improvement in the value. In such a way, the candidate policy converges to be optimal gradually.

Some modifications for value iteration and policy iteration are in [80].

2.2.1.2 Online Solutions of POMDP

The offline solutions of POMDP can guarantee the global optimality, and can be fast to execute. However, the comprehensive enumeration of the whole belief space means that the policy planning stage can take very long, which may be hours or days [81]. In a robot search problem studied in [82], even after simplifying the state space to be with around 7000 states, the offline planning still takes 20 mins. If during the execution, there is any environmental changes which necessitate the policy to be re-planned, it will be very computationally expensive. The online solution of POMDP is an alternative which can circumvent such problem.

The online planning only focuses on the search tree rooted at the current belief state, which contains the state space reachable within the time horizon. This largely reduces

the size of the search tree, and makes it possible to re-plan the policy during execution. Since only local and current information is considered, the planned policy is only valid from the current state till the end of time horizon, thus the planning stage and the execution state should interleave with each other. In such a way, if any environmental change is detected, it can soon be included into the new policy. We can see that the online Solution is a trade-off between the global optimality and the adaptability.

There are tree main categories of online Solutions of POMDP: branch-and-bound, Monte-Carlo Sampling, and heuristic search [81].

Branch-and-bound [83, 84] is the method of estimating the upper and lower bound of the value of a branch in search tree, and pruning the branches which do not worth exploring. The lower and upper bound of the value of a leaf node in the search tree can be approximated by offline planning. The bounds can then be propagated backwards to higher nodes, and the bounds of the higher branches can be obtained eventually in such a way. Monte-Carlo method [85] samples the nodes in the search tree to expand, and uses the sampled nodes to approximate the search tree. It also uses a Particle Filter to update the belief state in each simulation. It treats the system model as a black box with input and output, and a policy can be evaluated by doing Monte-Carlo simulations of it interacting with such black box. In such way, the value of a sampled node can be estimated. heuristic search evaluates the value of a leaf node with a heuristic function, and expands the search tree from the nodes of the most relevant reachable beliefs. Such beliefs are chosen by the criteria that they allow the search algorithm to make good decisions as quickly as possible [81].

Some other methods of approximating the search tree are introduced in [86].

2.2.2 Partially Observable Stochastic Game and Decentralized Partially Observable Markov Decision Process

The POMDP can be a framework of solving the planning and acting of a single agent in a partially observable stochastic environment. In some scenarios, there may be more than one intelligent agent involved in the operation. Such as when the targets can actively plan its action and evade from the pursuer, or when multiple pursuers are cooperating for the same mission, so that every player (a pursuer or an evader) in the game can do planning for its own interest. In such case, each planning should take into account of the possible knowledge and actions of the other players. The search and evasion game between the pursuer and the evaders can be constructed as a Partially Observable Stochastic Game (POSG), where at least one side in the game has incomplete and/or imperfect sensing of the world, and both side plan with conflicting goal. The cooperation between multiple

pursuers can be built as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), where every agent has only partial observation of the environment, but cooperate with a common goal. Although these are two different scenarios, but the underlying problems, the POSG and the Dec-POMDP, are very similar and only differ in whether the goal is shared. So both problems are introduced together in this Section.

The POSG can be described in a tuple $\langle I, S, \{A_i : i \in I\}, \{\Omega_i : i \in I\}, T, \{O_i : i \in I\}, \{R_i : i \in I\}\rangle$ [87], where

1. I is a finite set of players. i is the label of a certain player, and $i \in I$;
2. S is a finite set of states of the world;
3. A_i is a finite set of actions of the player i;
4. $T : S \times \{A_i : i \in I\} \longrightarrow p(S)$ is the state-transition function, mapping from a previous world state and the joint actions of all the players, to a probability distribution of next world states;
5. $R_i : \boldsymbol{S} \times A_i \longrightarrow R_i$ is the reward function, mapping from a current world state and a player action to a immediate reward to that player;
6. Ω_i is a finite set of observation of the player i;
7. $O_i : \boldsymbol{S} \times A_i \longrightarrow p(\Omega_i)$ is the observation function, mapping from a current world state and an action of the player i, to a probability distribution of the observations of the player i.

The model of Dec-POMDP is the same as the POSG, except for that the reward function R is shared among all the players.

The same as POMDP, the objective value function of a player i can also be formulated as the expected reward within the time horizon.

$$V_i = E\{\sum_{t=t_0}^{t_f} R_i(s_t, a_t^i)\} \tag{2.2}$$

Let Y_t^i to be the history of observation of player i, then $a_t^i = \pi^i(Y_t^i)$ is the policy of player i. $\delta = \{\pi^i : i \in I\}$ is the joint policy of all players, and δ_{-i} is the joint policy of all players except for i.

For the POSG with individual reward for each player, the solution should be a Nash Equilibrium δ^* [88], where

$$V_i|\{\delta^*_{-i}, \pi^{i*}\} \geq V_i|\{\delta^*_{-i}, \pi^i\},\ \forall \pi^i, i \in I \tag{2.3}$$

For the Dec-POMDP with shared reward, the solution is the joint policy which maximize the common value [89]:

$$\delta^* = argmax_\delta V|\delta \tag{2.4}$$

The same as POMDP, the POSG and Dec-POMDP can be solved offline or online. On top of POMDP, the POSG and Dec-POMDP contains multiple agent which have their plannings coupled, thus is much more complicated to solve. According to [32], solving Dec-POMDP is NEXP-Complete, which needs double exponential time in the worst case [89]. Thus simplification is also necessary in practically solving both problems.

2.2.2.1 Offline Solutions of POSG and Dec-POMDP

[89] has made a comprehensive survey of the offline solutions of POSG and Dec-POMDP. The optimal solution of POSG and Dec-POMDP is firstly developed in [87]. In [87], a dynamic programming method was designed, which consists of two steps: exhaustive backup and pruning dominated policy trees. In every iteration, the policy trees are built bottom up with exhaustive backup for one step, and the dominated policy trees are pruned to facilitate the further dynamic programming. However, in the example of [87], this algorithm runs out of memory after 4th iterations, because of the policy trees stored in the exhaustive backup.

A simplified method of dynamic programming is heuristic search method [90]. Similar to the heuristic search in the POMDP, the policy tree is built from top-down. The value of the nodes beyond immediate time step is estimated by a heuristic function, which completes the approximate value of the current branch. The nodes in the search tree are expanded in a best first manner.

Except for dynamic programming, the policy iteration method can also be applied on the POSG and Dec-POMDP [89]. Other approximate solutions are introduced in [89] as well.

2.2.2.2 Online Solutions of POSG and Dec-POMDP

The offline solution of POSG or Dec-POMDP can be fast and comprehensive during execution. Nevertheless, it still has the limitations shared among offline planning approaches: not being flexible or adaptive in a changing environment, because of the long planning time. So the online solution can be applied in the scenario which requires frequent re-planning.

However, there is not many works on the online solutions of POSG or Dec-POMDP [69, 91, 92]. This is because of the difficulty of having a compact representation of the multi-agent belief states without call back of the whole sensing history [89]. In the POMDP, the agent can maintain a belief state, which is a probability distribution of the state space and can fully represent the sensing history. In the POSG and Dec-POMDP, the belief of a player should also include the estimation of the policies of other players [87, 89]. So far, however, no state estimator function without perfect recall has been proposed, and no compact explicit representation of belief states for multi-agent settings has been introduced [89]. In the offline solutions such as [87], although the belief states are not described explicitly, it has been implicitly included in the sensing history, which is equivalent to all the information available to the agent. In I-POMDP [93], which is an other representation of multi-agent planning, the belief states are explicitly formulated. But this will induce an intractable infinite nesting: agent 1 may know the information held by agent 2, and agent 2 knows what does agent 1 knows, and agent 1 knows that agent 2 knows what agent 1 knows and so on.

During online planning and execution, it is impossible for an agent to store its whole history of observation. Having a compact belief state to fully represent all the useful information is crucial to the game playing. In [94], the author proved that under some conditions, such infinite nesting of belief can be replaced by a commonly held prior. However, this requires all the agents use the common knowledge to update such prior, and the agents should always plan the same set of joint policies. These requirements may be possible in a cooperative robot team, but are difficult to be satisfied in a non-cooperative game playing. Thus in other works, the agent either communicate to share the sensing history rather than holding belief state [92], or use heuristics to approximate the belief about other agents [91].

To the author's knowledge, there has not been works on the online solution of POSG for non-cooperative game playing, which also implies the difficulty of online solutions of POSG and Dec-POMDP.

2.3 Conclusion

In this chapter, the researches on different scenarios of robot search and pursuit evasion are introduced from different aspects. In a task-oriented perspective, the scenarios are categorized into search, monitoring, and multi-robot cooperation. The problem set-up, such as the characteristics of the environment, the numbers of the pursuers and the targets, and the sensing ability and dynamics of the players, defines each scenario and determines the suitable solutions. After reviewing these related works, a broad overview of the problem of robot search and pursuit evasion is provided. The separation of search and monitoring is suggested from this background, from the aspects of different prior knowledge about the targets, different goals to achieve on the targets, and different solutions. This thesis was inspired by such separation. The problem formulation will be done in Chapter 3, which is developed based on the background from this chapter.

From the reviewed works, some underlying technical details are then picked up, whch are the POMDP and POSG/Dec-POMDP. These are the main mathematical problems to solve, and are the potential methodologies for us. The scope is not limited on only robot search and pursuit evasion, but also on the theoretical solutions on these problems. This Section gives a basic concept of the solutions, together with the difficulties to face such as the computation efficiency and scalability. With such concept in mind, throughout the whole of this work, balancing performance and practicability is the main focus on developing solutions. In Chapter 4, 5 and 6, different solutions will be proposed, which are innovatively developed to be practical to implement and have advantage over baseline methods.

Chapter 3

Problem Formulation of Simultaneous Search and Monitoring

Based on different kinds of problem formulation introduced in Chapter 2, this chapter first does the basic modelling of the environment, the pursuers, and the targets. Such modelling tries to be non-specific and flexible, to allow different problems to be studied on it. Section 3.1 define the arena of the search and pursuit evasion between the pursuers and the targets. Section 3.2 and 3.3 defines the model of motion, communication, and sensing of the pursuers and the targets. The models will be applied throughout the rest of this thesis.

From the ideas of search and monitoring, which appears to be independent tasks, the concept of simultaneous search and monitoring (SSM) is developed and defined informally in Section 3.4. The SSM in different scenarios be defined in detail and solved in Chapter 4, 5 and 6.

3.1 Modelling of the Environment

As mentioned in the Introduction, this thesis focus on research in a big outdoor open space. It is assumed that the search and pursuit evasion happens in a confined environment, which is a $L \times W$ rectangle area. The terrain or occlusion is not considered in this thesis, which makes it a fully connected and homogeneous space. To facilitate numeric computations over such environment, a grid network is used to represent the space. For a $L \times W$ rectangle area ς, it is discretized into a grid $\varsigma = \{c_{i,j} : i = 1, 2, ..., n_x, j = 1, ..., n_y\}$,

where $c_{i,j}$ denotes grid cells and i, j denote the coordinates on the x and y axis. Let $N(c_{i,j}) = \{c_{i',j'} : i' \in [i-1, i+1], j' \in [j-1, j+1], c_{i',j'} \in \varsigma\}$ be the set of neighbouring cells of $c_{i,j}$. Two cells are *connected* on the grid when they are the neighbouring cells of each other. The grid and the connection between cells are illustrated in Figure 3.1, where each black points denote the centre of a cell and the lines are the connection between neighbouring cells.

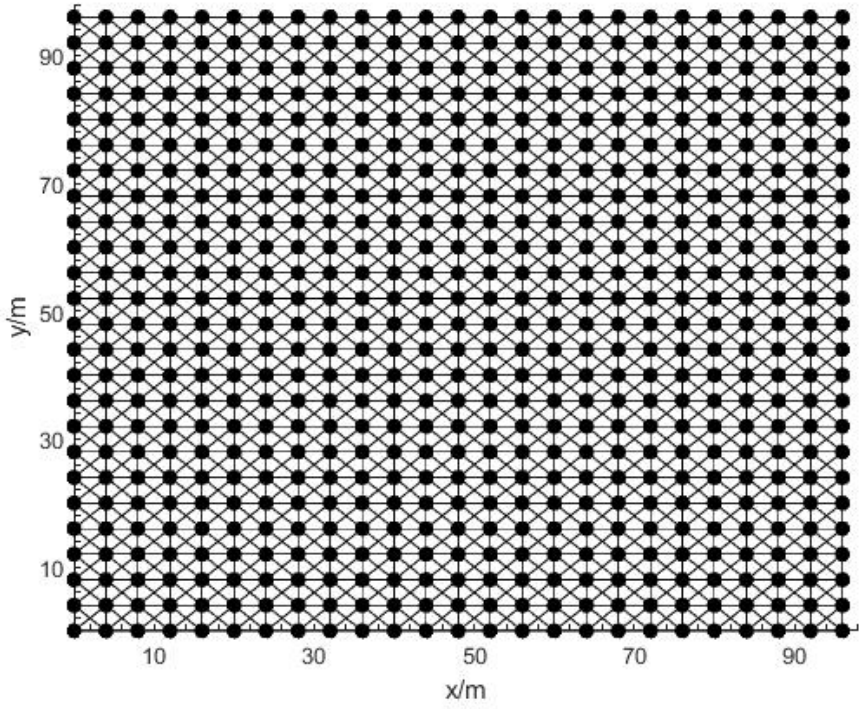


FIGURE 3.1: grid map of the environment

There can be n ground targets and m aerial pursuers sparsely scattered in ς. Each target is distinguishable and is assigned with an ID $\lambda \in \Lambda$. Λ is the set of all the targets. The ID ρ and set Γ is defined for the pursuers, respectively. Each pursuer or target can be located in a certain cell, and can move between *connected* cells. In the remaining part of the thesis, the term agent is used interchangeably with pursuer, and the term evader is used interchangeably with target.

3.2 Modelling of the Pursuers

3.2.1 Motion Model of the Pursuers

For search and pursuit evasion about a aerial pursuer in a outdoor environment, the Dubins Vehicle model is usually applied for the aerial vehicle [54, 57, 59, 60]. In such a

model, the control inputs of the agent are the accelerations of its forward velocity and heading, which are bounded. This model can take into account the inertia of the vehicle, thus including the limitations on the ability of the agent to change its motion, such as minimum turning radius. Nonetheless, in this thesis, it is assumed that the environment is big and the distribution of targets are sparse. Thus the influence of minimum turning radius on the effort of search and monitoring should be negligible. The inertia of the agent is ignored, and it is assumed that the agent can follow an arbitrary trajectory within a speed bound V_p. Let $x_\rho(t)$ denote the location of agent ρ at time t.

Although in the main part of this work, the manoeuvrability of the agent is not considered. However, in the end of Chapter 4, 5 and 6, the scenarios where the agent model is a dubin vehicle will be studied. This is to validate the feasibility of this work in a more realistic system.

3.2.2 Sensing Model of the Pursuers

For the sensing of the agent, it is assumed that each aerial pursuer carries a target sensor, which is mounted a gimbal and keeps looking downwards to the ground. It is assumed that the sensor footprint is a rectangle and does not rotate on the ground. For a pursuer ρ in grid cell $c_{i,j}$, its sensor footprint is the area $\Delta_\rho = \{c_{i+a,j+b} : a, b \in \{-k, -k+1, ..., 0, ..., k\}\}$, which is a $(2k+1)$ by $(2k+1)$ square centred at $c_{i,j}$. Let $\Delta = \{\Delta_\rho : \rho \in \Gamma\}$, which is the total sensor footprint. Without losing generality, it is assumed that $(2k+1) = n_x/L, (2k+1) = n_y/M$, where N and M can be any positive integers. Thus if agent visits cells $C_s = \{c_{(2k+1)l-k,(2k+1)m-k} : l = 1, 2, ..., L, m = 1, 2, ..., M\}$, the whole environment can be swept by sensor footprint. Such assumption reduces the number of cells to consider, when planning about searching in the whole environment.

Let y_t^ρ be the measurement of pursuer ρ at time t. $p(y_t^\rho|c)$ denotes the probability function of sensing, indicating the probability of possible individual measurement y_t^ρ, given that the target is at c at time t. Thus

$$p(y_t^\rho|c) = \begin{cases} p(1|0) & \text{false positive} \\ p(0|0) & \text{true negative} \\ p(0|1) & \text{false negative} \\ p(1|1) & \text{true positive} \end{cases} \tag{3.1}$$

3.2.3 Communication Model of the Pursuers

This work does not consider the aspect of distributed sensor fusion, or how to maintain a tree structure for communication. It is assumed that the agents can not only sense the location of each other perfectly, but also have a communication scheme which can share all their sensing in real time. Then, the set of agent locations $\{x_\rho(t) : \rho \in \Gamma\}$ are perfectly shared within agents. This is not a strict assumption, for several reasons. Firstly, it should be much easier to detect aerial teammates than to spot camouflaged ground targets. because a agent can broadcast its own location, or carry some signal trackers for the teammates to measure; Secondly, the agents only need to communicate about measurements when a detection happens. Thus when the targets are sparse, it does not require high communication bandwidth. With such assumption, it is assumed that all the agents share the same knowledge of the environment, then the measurement and estimation of different agents are not differentiated. Let y_t denote the joint measurement of all the agents at time t, which is known by all the pursuers.

However, in the last part of the thesis, which is Section 6.5, the range limit on the communication will be considered. This is to test the practicability of this research in a non-ideal circumstance, and also to improve the scalability of the cooperation between agents.

3.3 Modelling of the Targets

As defined in Section 3.2.2, the sensor footprint of the agent cannot cover the whole environment, and may produce false measurements. Therefore the information of the targets can not be perfectly sensed at all times. A mathematical model needs to be built for the target, in order to estimate and predict target locations, given partial and imperfect observations. In related works, there are mainly three kinds of target models: Gaussian uncertainty model, awareness model, and probability distribution model.

Gaussian Uncertainty Model

In this model, the target is a known base model perturbed by a zero mean Gaussian noise. The observation is perturbed by a zero mean Gaussian noise as well. The estimation of target state and the posterior covariance matrix are updated by Kalman Filter.

In [53], the target model is ignored. The sensing noise are all Gaussian, and the sensing range is infinite. In [59, 60], the target model is known, but the model input is a Gaussian noise. In [61], the target model is known and with a known input, but is perturbed by a Gaussian noise. These models assume that the target follows a pattern of motion which

is known a priori, but with a Gaussian perturbation. However, in this thesis, the target motion can be very unpredictable, or even adversarial against the effort of pursuer, thus an assumed pattern of target dynamics can not accurately describe and estimate target information.

Awareness Model

An alternative of modelling the target is not to consider the specific target information, but to model the certainty of the information in general. A empirical model is built to describe what is called the *awareness* level of the environment at every location [46, 47]. The *awareness* level of certain area grows with search efforts on it, and declines when left alone. This model does not deal with detailed information of target location, therefore can not be applied in cases which require accuracy.

Probabilistic Distribution Model

The probability distribution model is a very commonly adopted framework, which can combine the benefit of the accuracy of estimating a specific target location, and the generality of dealing with uncertainties. In this kind of model, the target position is modelled as a probabilistic distribution. The distribution evolve according to a target dynamic model, and can be updated by measurement, using Bayesian formulation. This model can precisely estimate and predict the probability of target presence. Besides, it is of a more general form, which can incorporate different kind of target dynamics, road map, and sensor model, thus can be more flexible to address different problems. Examples of the works which applied probability distribution map are [20, 22, 23].

In this thesis, two kinds of targets are studied separately: the randomly moving targets and the evasive targets. According to the the above discussion, the probability distribution model is chosen to model the randomly moving targets, which is the most suitable in this case. Some modification of such model will be made, to represent the evasive targets.

3.3.1 Modelling of Randomly Moving Targets

3.3.1.1 Motion Model of the Targets

Let $x_\lambda(t)$ denote the location of target λ at time t. In the discretized environment, at each time step, each target may move from its current cell to a neighbouring cell, or stay unmoved. Assume that the target motion is a Markov Chain, which is a commonly applied model [20–22]. Then the target movement only depends on its current location,

and is not related to its history. Let $p(c|c')$ be the transition function, representing the probability of target moving from c' to c in a time step, where

$$p(c|c') = \begin{cases} p_s & \text{if } c = c' \\ p_{c|c'} & \text{if } c \in N(c') \\ 0 & \text{else} \end{cases} \tag{3.2}$$

$p_{c|c'}$ is the probability that the target moves from c' to a neighbouring cell $c \in N(c')$. p_s is the probability that target stays unmoved. $p_s + \sum_{c \in N(c')} p_{c|c'} = 1$. Let $N_1(c')$ denote the neighbouring cells which are in the lateral or longitudinal directions with c', and let $N_2(c')$ denote the neighbouring cells in the diagonal directions with c'. Assume $p_{c|c'}$ is proportional to the inversion of the distance between c and c'. Then it can be obtained that

$$p_{c|c'} = \begin{cases} \frac{(1-p_s)}{|N_1(c')|+\sqrt{2}|N_2(c')|} & \text{if } c \in N_1(c') \\ \frac{\sqrt{2}}{2}\frac{(1-p_s)}{|N_1(c')|+\sqrt{2}|N_2(c')|} & \text{if } c \in N_2(c') \end{cases} \tag{3.3}$$

where $|.|$ is a operator to get the size of a set.

3.3.1.2 Estimation Model of the Targets

Assume that the environment is so large that the union of the agent sensor footprints can not cover the whole environment, and there are possibly false measurements. Therefore, the target information is partially observable to the agents, and every agent keeps hold of a probability distribution map of each target. Let $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$ be the estimated probability of target λ being in cell c at time t, given Y_t which is the set of measurement up to time t. Because the agents share the sensing information, $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$ is known by all agent with no difference.

After holding a initial probability distribution of the targets, the agents can update or predict the probability distribution, by utilizing the motion model of the targets and the sensing model of the agents. Bayesian formulation is applied for the update of $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$, which is based on the work of [20–22]:

1. Prediction. Compute prediction using the prior probability distribution $\hat{\mathrm{P}}_\lambda(c', t-1|Y_{t-1})$, the transition function (3.2), and the Chapman-Kolmogorov equation

$$\hat{\mathrm{P}}_\lambda(c, t|Y_{t-1}) = \sum_{c' \in \varsigma} p(c|c')\hat{\mathrm{P}}_\lambda(c', t-1|Y_{t-1}) \tag{3.4}$$

2. Correction by observation. Update the prediction for cells which are being observed, using Bayes' theorem

$$\hat{\mathrm{P}}_\lambda(c,t|Y_t) = \frac{\hat{\mathrm{P}}_\lambda(c,t|Y_{t-1})p(y_t|c)}{\sum_{c'\in\Delta}\hat{\mathrm{P}}_\lambda(c',t|Y_{t-1})p(y_t|c')} \tag{3.5}$$

3. Correction by inference. For cells outside of sensor footprint, the prediction can be corrected using the fact that $\sum_{c\in\varsigma}\hat{\mathrm{P}}_\lambda(c,t|Y_t)dc = 1$

$$\hat{\mathrm{P}}_\lambda(c,t|Y_t) = \hat{\mathrm{P}}_\lambda(c,t|Y_{t-1})\frac{1-\sum_{c'\in\Delta}\hat{\mathrm{P}}_\lambda(c',t|Y_t)}{\sum_{c'\in\varsigma/\Delta}\hat{\mathrm{P}}_\lambda(c',t|Y_{t-1})} \tag{3.6}$$

The Probability map $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ can provide a general description of the target location. For the targets whose information is highly uncertain to the agents, such distribution can conveniently provide the probability of detection for each search plan of the agents. However, for the targets whose locations are relatively certain, more specific representation of information is needed to facilitate more precise actions. To separate generic or precise operations on different targets, the targets are categorized as known or unknown. Target λ is viewed as known, if the aggregation level of $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ becomes higher than a upper threshold, which is normally when it is detected. $\hat{\mathrm{x}}_\lambda(t)$ is the estimation of target location $x_\lambda(t)$, which is calculated from $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$, or it can be simplified to be the location where it was detected for the last time. Let $\Lambda_t \in \Lambda$ denote the set of known targets at time t. If the aggregation of $\hat{\mathrm{P}}_\lambda(c,t|Y_t)$ is lower than a bottom threshold, or if a agent fails to detect λ when it traverses $\hat{\mathrm{x}}_\lambda(t)$, λ is lost and becomes unknown. Each known target is said to be under monitoring until lost.

For all the unknown targets $\lambda \in \Lambda/\Lambda_t$, let $\hat{\mathrm{P}}_u(c,t|Y_t)$ be their total probability distribution, thus

$$\hat{\mathrm{P}}_u(c,t|Y_t) = \sum_{\lambda\in\Lambda/\Lambda_t}\hat{\mathrm{P}}_\lambda(c,t|Y_t) \tag{3.7}$$

With the total probability distribution of unknown target, the likelihood of detecting a hidden target can be determined for a certain path plan of the agent, thus making it easy for the planning of search mission. The location estimation $\hat{\mathrm{x}}_\lambda(t)$ of a known target can allow the agent to plan a more specific trajectory to monitor that target.

The whole environment with the randomly moving targets is shown in Figure 3.2. The space is partitioned by grid cells. Circles denote the targets, among which the filled ones are known targets. The rectangles are the agent sensor footprints. The crosses are the estimated locations of known targets. The numbers label the target IDs and the

numbers with underline label the agent IDs. The plus signs denote cells in C_s. The contour denotes the total probability distribution of unknown targets

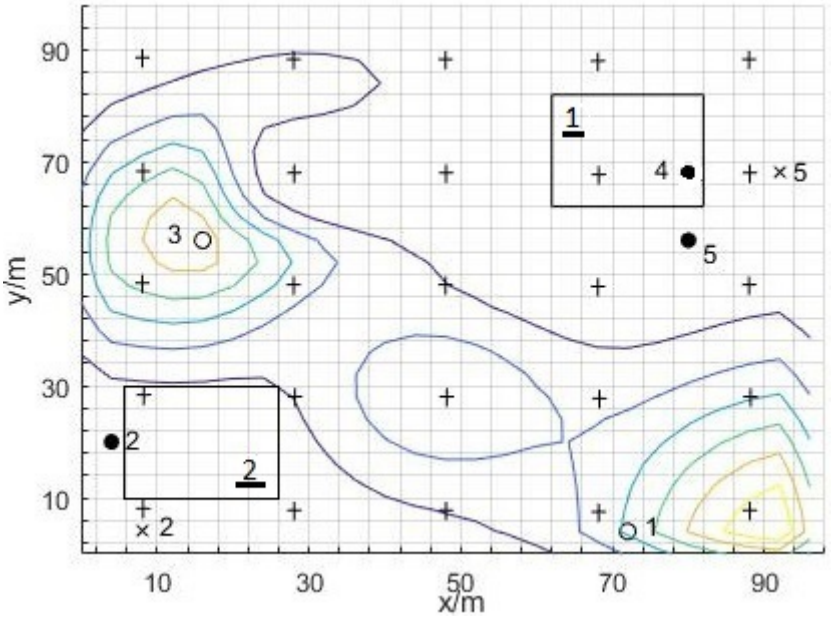


FIGURE 3.2: example of the environment with randomly moving targets

3.3.2 Modelling of Evasive Targets

3.3.2.1 Motion Model of the Targets

Let $x_\lambda(t)$ denote the location of the evasive target λ at time t. The targets can move between neighbouring cells or stay unmoved. Compared with the randomly moving targets, the difference is that the evasive targets can proactively plan its own strategy and choose its actions, which should be adversarial to the effort of the pursuers. These plans and motions are normally not know a priori to the agents. Thus there should not be a specific pattern of target motions to define. Instead, a worst case model is used to describe the possible motions of the target.

Let V_t be the speed limit for the evaders, then it can be said that if a target moves in speed V_t, it can reach a neighbouring cell in T_x time steps. $\theta(c|c', t)$ is the transition function representing whether a target can move from c' to c at time t, and

$$\theta(c|c',t) = \begin{cases} 1 & \text{if } c = c' \\ 1 & \text{if } c \in N(c'), \text{ and } t = i * T_x \\ 0 & \text{else} \end{cases} \tag{3.8}$$

where i is any integer.

This model contains all possible motions of the evader at certain time step. The grid map defined in Section 3.1 contains the directions on the lateral, longitudinal, and diagonal directions. The motion of the target should be omnidirectional, but the distance between neighbouring cells are different on each direction. Hence, it is very difficult to represent the realistic worst case assumption of the target motion using the Markov-Chain-like model defined in Equation (3.8), because it takes different time steps to arrive in a different neighbour. To simplify the estimation and planning in the scenario of the evasive targets, it is assumed that the target only moves on the lateral and horizontal directions. By having a smaller value of T_x, the chance of diagonal motion can be included in the worst case model.

3.3.2.2 Estimation Model of the Targets

Similar to the case of randomly moving targets, a distribution map is used to estimate the possible locations of the evasive targets. Nevertheless, rather than using probabilistic model, a occupancy grid map $\hat{\mathrm{M}}_\lambda(c, t|Y_t)$ is used to represent all the possible locations of a target, thus to estimate the worst case evader behaviours. $\hat{\mathrm{M}}_\lambda(c, t|Y_t) = 1$ if and only if target λ is possibly in cell c at time t, given $Y_t = \{y_t, y_{t-1}, ..., y_0\}$ which is the set of pursuer measurements of up to time t.

To make it compatible with the worst case model, the sensing model defined in Equation 3.1 is transformed to be Equation 3.9. $\phi(c|y_t)$ denotes result of measurement, indicating whether a target may be at c, given the current measurement y_t.

$$\phi(c|y_t) = \begin{cases} 0 & \text{negative} \\ 1 & \text{positive} \end{cases} \tag{3.9}$$

This sensing model includes the false positive into the worst case target presence, but ignores the false negative for the ease of analysing. The evolution of the occupancy grid map can then be defined, similar to that of the randomly moving targets.

1. *Prediction.* Compute the prediction using the prior map $\hat{\mathrm{M}}_\lambda(c, t-1|Y_{t-1})$ and the transition function (3.8)

$$\hat{\mathrm{M}}_\lambda(c, t|Y_{t-1}) = sign(\sum_{c' \in \varsigma} \theta(c|c', t)\hat{\mathrm{M}}_\lambda(c', t-1|Y_{t-1})) \tag{3.10}$$

2. *Correction by observation.* Update the prediction for cells which are being observed, with the sensory model

$$\forall c \in \Delta, \hat{\mathrm{M}}_\lambda(c,t|Y_t) = \phi(c|y_t) \tag{3.11}$$

3. *Correction by inference.* If a target is detected, set the map outside the sensor footprint to be zero. It utilizes the fact that $\sum_{c\in\Delta} \hat{\mathrm{M}}_\lambda(c,t|Y_t) = 1$ if λ is detected, or $\sum_{c\in\Delta} \hat{\mathrm{M}}_\lambda(c,t|Y_t) = 0$ if λ is outside sensor footprint.

$$\forall c \notin \Delta, \hat{\mathrm{M}}_\lambda(c,t|Y_t) = \hat{\mathrm{M}}_\lambda(c,t|Y_{t-1})(1 - \sum_{c'\in\Delta} \hat{\mathrm{M}}_\lambda(c',t|Y_t)) \tag{3.12}$$

The operations on the worst case occupancy grid map will focus more on clearing areas possibly occupied by the target, and prevent the recontamination of the cleared area. Therefore, for the evasive targets, the estimation of the specific target location is not needed. The whole environment with the evasive targets is shown in Figure 3.3. The space is partitioned by grid cells. The circles denote the targets. The numbers label the target IDs and the numbers with underline label the agent IDs. The rectangle is the pursuer sensor footprint. Each blue polygon denotes the occupancy grid map of a target.

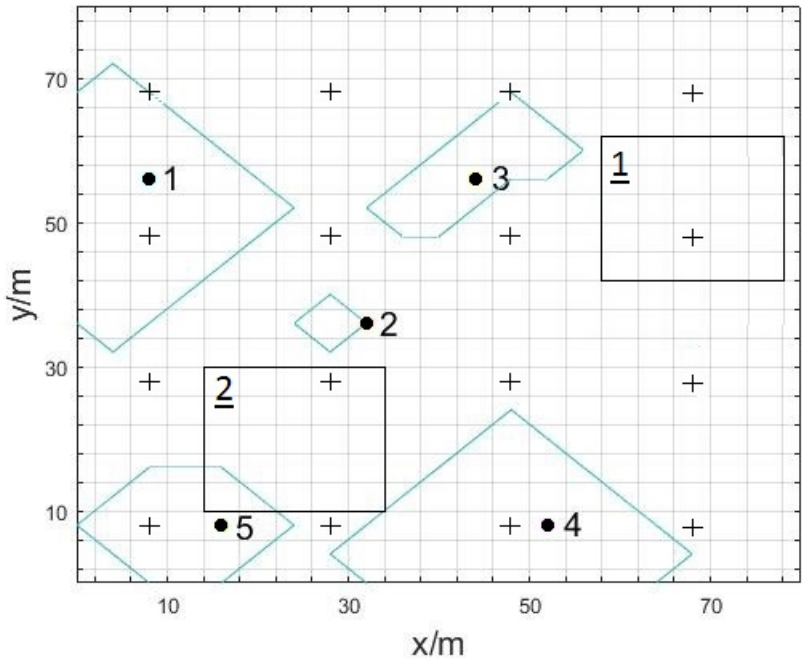


FIGURE 3.3: example of the environment with evasive targets

3.4 Concept of Simultaneous Search and Monitoring

Given the models of the environment and the players, this section now informally formulate the concept for simultaneous search and monitoring. In Section 3.4.1, Such concept is defined based on a description of uncertainty. Such representation of uncertainty will be defined in detail in Section 3.4.2 and 3.4.3, each for either randomly moving targets or evasive targets.

3.4.1 Simultaneous Search and Monitoring to Reduce Uncertainty

As reviewed in Chapter 2, the search and monitoring problems are normally studied in isolation. The robot search is to detect hidden targets to obtain specific locational information about them, the mission finishes when the targets are found; the robot monitoring is to surveil the known targets to update their location, it is normally a dynamic process. The prior knowledge of the targets are different in both missions, and the goals are different too. Thus the solutions proposed for the two problems can be very different. That is why these two problems are normally seen as unrelated. However, as introduced in Section 1.1, the search and monitoring are both needed at the same time. If the agents only focus on one mission, either the detected targets will run away, or the hidden targets may not be found.

It appears that search and monitoring are two independent missions, which compete for the effort of the pursuer. A straightforward way of reconciling search and monitoring is to quantify the goal of the two missions, and add them up to a overall goal. A task assignment method can be implemented by scheduling the plan of the agent to do the search and monitoring in turn, thus to achieve a higher overall goal by having a trade-off [3]. However, a target can change between known and unknown, making search and monitoring inherently connected. A hidden target becomes known with certainty after being detected in a search mission, it should be sensible to add it under monitoring. A known target may get lost during monitoring, and should then be considered in a search mission. If treating the two missions independently, in a plan, a target which changes its status will not subsequently be attended. Thus this isolation can not accurately represent the practical need of SSM, and does not enable the synergy between search and monitoring.

Therefore, the combination of search and monitoring should be treated as a cooperation, to better react to the possible contingencies happening on a target. We do this by defining a united representation of uncertainty for each target, and letting the simultaneous search and monitoring (SSM) to obtain and update the information of mobile targets

in a dynamic way, to maintain a low overall uncertainty. In such way, the agents can reduce the overall uncertainty by searching for unknown targets and monitoring known targets at the same time, and also make proper plan about the contingencies of a target being detected or lost.

For a target λ, let Un_λ denote the uncertainty about its location, which is hold by the pursuers. Let $\sum_{\lambda \in \Lambda} Un_\lambda$ be the total uncertainty about all targets. The goal of the SSM is minimizing the total uncertainty $\sum_{\lambda \in \Lambda} Un_\lambda$. Un_λ will then defined in detail for randomly moving or evasive targets.

3.4.2 Uncertainty Representation of Randomly Moving Targets

3.4.2.1 Combining Search and Monitoring with a United Uncertainty Representation

In the review of related works in Chapter 2, the works on robot search and pursuit evasion have been classified into two main categories: search and monitoring. In the work about search, the targets are normally modelled as a probability distribution. In the works about monitoring, the targets are modelled as a known location with possible uncertainties around it. In the modelling of randomly moving targets in Section 3.3.1, both methods are taken. A probability distribution is used to describe each target, but some targets with higher certainty are classified as known and their specific locations are recorded.

In the first glance, having only the probability distribution of every target is enough to represent the uncertainty. The higher aggregation of probability distribution, which is like a spike, should mean that the knowledge of this target is certain. A lower aggregation, which is a relatively flat distribution, means higher uncertainty. However, the disadvantage of having probability distribution is a low computational efficiency. When planning with such model, all locations in the environment may possibly be considered, because they are treated in a general way. In the works about monitoring multiple mobile targets, the target model normally consists of a known location plus its uncertainty. With such model, there is only one location to consider for each target, thus can significantly reduce the difficulty of the planning for monitoring. In [54], the authors take advantage of the upper bound of the target speed, and obtain a time limit, so that if a pursuer intermittently visits the last detected location of a target within this time limit, the target will be guaranteed to be re-detected. In [59–61], such model for known targets is also applied, to simplify the path planning. This is the reason why in this thesis, the targets are differentiated as known and unknown, and different models are used for them.

To formulate a combined search and monitoring mission, this work takes advantage of the connection between both missions. The total certainties of the known targets are taken as the total uncertainty of all the targets. And the uncertainties of the unknown targets are ignored. This is for two reasons. Firstly, for most of applications, compared with general information of unknown targets, the knowledge of known targets is more specific and can provide more practical utility. Examples of these applications can be sea rescue of a ship crew, search and capture of criminals, etc.. Secondly, as mentioned above, the targets can change status during a mission, thus the unknown targets are considered implicitly. The total certainty can be contributed by updating the information a known target, or detecting a new target and put it under monitoring. In such a way, the search and monitoring are combined in a united goal.

3.4.2.2 Uncertainty Representation of Known Targets

In the Section 3.3.2.2, it has defined $\hat{\mathrm{x}}_\lambda(t)$ as the estimated location of a known target. Then, the uncertainty of such estimated location is defined as follows:

Definition 3.1. For the estimated location $\hat{\mathrm{x}}_\lambda(t)$ of a known target $\lambda \in \Lambda_t$, the *belief probability* is the probability that its actual location $x_\lambda(t)$ is within $F(\hat{\mathrm{x}}_\lambda(t)) = \{c_{i+a,j+b} : a, b \in \{-k, -k+1, ..., 0, ..., k\}, c_{i,j} = \hat{\mathrm{x}}_\lambda(t)\}$. The *belief probability* is denoted by $\tilde{B}_\lambda$. $F(\hat{\mathrm{x}}_\lambda(t))$ is area of the same shape with the sensor footprint of the pursuer, and is centred at $\hat{\mathrm{x}}_\lambda(t)$.

The rationale of $\tilde{B}_\lambda$ is that, it provides a lower bound of the probability of target λ to be re-detected, if a agent visits $\hat{\mathrm{x}}_\lambda(t)$ at time t.

If a known target is not currently measured, its *belief probability* will degrade because of its random motion. A model will be defined, to estimate the evolution of *belief probability* when not being measured. Assume a target is initially placed at the centre of the environment, which is denoted as $c*$, and $\hat{\mathrm{x}}_\lambda(0) = c*$. Assume that this initial location is known for certain, then

$$\hat{\mathrm{P}}_\lambda(c, 0|Y_0) = \begin{cases} 1 & \text{if } c = c* \\ 0 & \text{else} \end{cases} \tag{3.13}$$

Let its probability distribution $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$ evolve on its own without measurement, based on the model in Section 3.3.1.2. The *belief probability* $\tilde{B}_\lambda$ at each time step is calculated, such that

$$\tilde{B}_\lambda = \sum_{c \in F(\hat{\mathrm{x}}_\lambda(t))} \hat{\mathrm{P}}_\lambda(c, t|Y_t) \tag{3.14}$$

The evolution of $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$ and the calculation of $\tilde{B}_\lambda$ is illustrated in Figure 3.4, where at every 0.2 second time step, the probability of target to stay unmoved is $p_s = 80\%$.

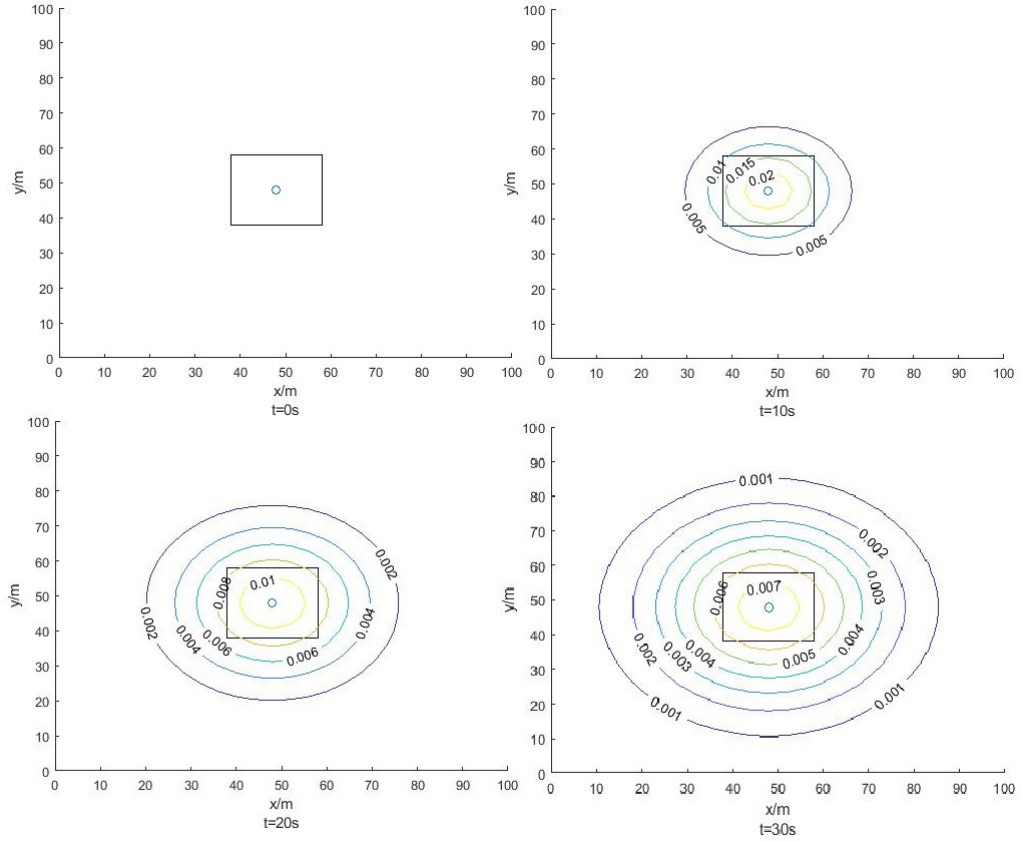


FIGURE 3.4: calculation of *belief probability*

Let $\tilde{B}^*_\lambda(t)$ denote the degrade function of the *belief probability* without measurement, which is shown in Figure 3.5 (a). In [54], by remembering the last detection location of a target and knowing its speed limit, the agent can guarantee the re-detection of a target by having a simple planning. It will also be shown here, that if classifying a recently detected targets as known and modelling it as a known location plus *belief uncertainty*, it can implicitly contain a benefit for re-detecting, thus facilitate the planning of monitoring. It has been shown that the *belief probability* is a lower bound of the probability of target λ to be re-detected. Thus we can see that, if a agent is to re-detect a target λ at time t' after the last detection, it will have a detection probability of at least $P = \tilde{B}^*_\lambda(t')$. If the agent leave the target after time t' and never comes back, the expected *belief probability* of target λ at each time instant is calculated as follows:

$$\tilde{B}_\lambda(t) = \begin{cases} \tilde{B}^*_\lambda(t) & \text{if } t \leq t' \\ \tilde{B}^*_\lambda(t') \times \tilde{B}^*_\lambda(t - t') & \text{if } t > t' \end{cases} \tag{3.15}$$

The functions of expected *belief probability* w.r.t future time, where there may be one or more attempts of re-detection, is shown in solid lines in Figure 3.5 and 3.6. They are compared with the expected *belief probability* without attempt of re-detection, which is $\tilde{B}^*_\lambda(t)$ and is shown in dash lines.

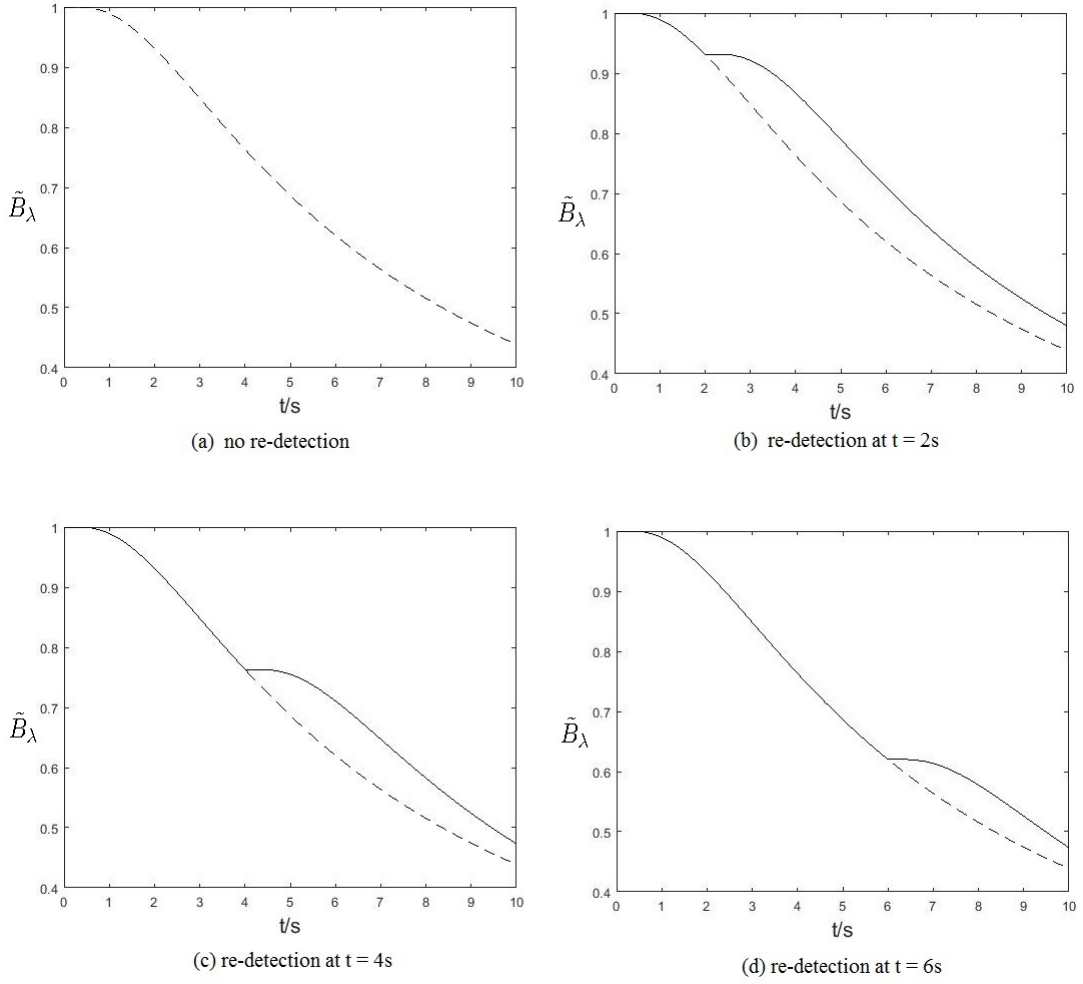
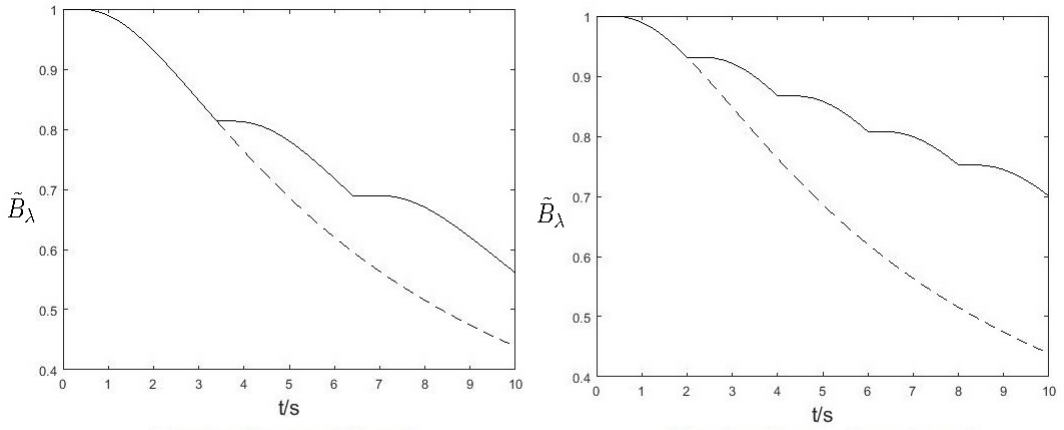


FIGURE 3.5: evolution of *belief probability* (1)

From Figure 3.5 (a) we can see that, when there is no measurement, the *belief probability* drops much slower in the first few seconds than later. This is because in the beginning of the time, although the target may move, its distribution is still concentrated around its original location and is mostly within the area of $F(\hat{\mathrm{x}}_\lambda(t))$.

It can also be seen that, if the agent tries to re-detect a known target, the more attempts it makes, the higher expectation of *belief probability* can be maintained. Although at



(a) re-detection at t = 3.33s and t = 6.67 s

(b) re-detection at t = 2s, t = 4s, t = 6s, and t = 8s

FIGURE 3.6: evolution of *belief probability* (2)

each attempt of re-detection, the expected *belief probability* $\tilde{B}_\lambda(t)$ will not be improved, cause the likelihood of successful detection is always around $\tilde{B}_\lambda(t)$. However, a successful detection can update the location of the target, which can make the future *belief probability* to drop slower compared with no-redetection, and make it easier for the future re-detection. This property of *belief probability* implicitly encourage the monitoring of known targets, and is very simple which does not require complicated calculation.

At last, the representation of the uncertainty for each target is defined as follows:

$$Un_\lambda = \begin{cases} 1 - \tilde{B}_\lambda & \text{if } \lambda \in \Lambda_t \\ 1 & \text{else} \end{cases} \tag{3.16}$$

With this definition, the agents can reduce total uncertainty by synergies of search and monitoring. The agents can search new targets to enlarge Λ_t, or to monitor known ones to increase $\tilde{B}_\lambda$.

3.4.3 Uncertainty Representation of Evasive Targets

The evasive targets do not follow a known stochastic motion pattern. Instead, they can take advantage of their perfect sensing, and plan the evasion to maximize its interest. Therefore, the idea of efficient search does not apply anymore. For a target which is found, it will try to run away from the previous location after being detected. So it does not improve the efficiency of monitoring, by categorizing recently found targets as known and try to re-detect it in its previous location. Hence, the evasive targets are not

differentiated as known and unknown. The model defined in Section 3.3.2.2 can be used to estimate the worst case distribution of the possible target locations.

Definition 3.2. For the agents, $\tilde{E}_\lambda$ denotes the *uncertainty* about target λ, which is the size of the occupancy grid map of λ. $\tilde{E}_\lambda(t) = \sum_{c \in \varsigma} \hat{\mathrm{M}}_\lambda(c, t|Y_t)$.

Let $Un_\lambda = \tilde{E}_\lambda$ to represent the uncertainty about a target in the scenario with evasive targets. Without the sensing of the agent, the occupancy grid map always grows as a result of the adversarial target behaviour. To reduce and limit the total uncertainty, the agents need to predict the growth of the occupancy grid maps, and plan a strategy to detect the targets or at least clear parts of their occupancy grid maps. This also achieves the simultaneous search and monitoring, by dynamically reducing the uncertainty of all targets at all times.

The definition of uncertainty for randomly moving and evasive targets will be utilized in Chapter 4, 5 and 6 in more detail, and the objective functions of the agents will also be defined formally to guide the strategy planning.

3.5 Conclusion

In this Chapter, the problem formulation is achieved by firstly setting up the environment and the players, and then proposing a abstract notion of simultaneous search and monitoring. The environment is discretized to allow a real-time planning. The motion, sensing and communication models of the pursuers and the targets are defined according to realistic applications. Given the partial observation of the pursuer, two estimation models are defined for the agents to conjecture and predict the location of either randomly moving or evasive target.

By analysing the connection between the unknown targets to be searched, and the known targets to be monitored, the gap between search and monitoring is found to be unnecessary. The concept of SSM is then developed, which is to reduce the total uncertainty of the dynamic targets. For the randomly moving targets, the targets with different uncertainties are classified and treated differently, to make the planning more efficient. The evasive targets are not divided, and are evaluated in a worst case estimation. The SSM of randomly moving targets with a single pursuer will be solved in Chapter 4. The SSM of evasive targets with a single pursuer will be solved in Chapter 5. The cooperative SSM of randomly moving or evasive targets will be solved in Chapter 6.

Although the perfect motion and communication models are assumed for the pursuers, they will be relaxed in later chapters. The Dubin Vehicle model for the aerial pursuer

will be considered and tested in Section 4.5.3, 5.6.3, and 6.6, for every scenario. The communication range limit will be considered in Section 6.5 and 6.6, for the cooperative SSM.

Chapter 4

Simultaneous Search and Monitoring of Randomly Moving Targets

The problem of simultaneous search and monitoring (SSM) is formulated in Chapter 3. It has been introduced in Chapter 1 that, the problem of SSM contains two main technical problems, the first is the sensor scheduling and strategy planning, and the second is multi-agent cooperation of the agents. To solve these problems, this thesis take an approach of splitting them into two steps: solving the SSM with a single pursuer, and build the cooperative SSM strategy on top of the solution for a single agent. Chapter 4 and 5 will be about the SSM between a single pursuer and multiple randomly moving targets or evasive targets. Chapter 6 will develop the cooperation of multiple agents for the same tasks.

This chapter first formulates the state space and objective value function of the problem of single pursuer SSM of randomly moving targets. A Partially Observable Markov Decision Process (POMDP) is built for this SSM. A novel concept of policy reconstruction is developed, to make it easy to incorporate heuristics into the policy planning. Online solutions are chosen instead of offline, to have a scalable and adaptive solution. Three online solutions are designed and compared. The conventional fixed sequence of action method is first devised, as the simplest and fastest solution. The hybrid of branching and fixed sequence of action method are developed secondly, which is the trade-off between the computational efficiency and optimality. A novel heuristic reactive policy reconstruction method is proposed in the last. The superiority of the heuristic reactive policy is proved theoretically and validated through simulation. The heuristic reactive

policy is chosen to be the solution of the single pursuer SSM of randomly moving targets, for its good performance and fast computation.

The Dubin Vehicle model of the UAV is considered in the last of the simulation. Which shows the practicability of this work.

4.1 Formulating the Partially Observable Markov Decision Process

For the single pursuer SSM of randomly moving targets, a formal problem statement can be done, based on the modelling in Chapter 3. Considering the fact that the motion of the random targets is a Markov Chain, and that the environment is partially observable to the agents, a POMDP framework is used to build this problem. The general structure of POMDP is introduced in Section 2.2.1. This model consists of an internal Markov Decision Process (MDP) and an observation model of the agent. Such model is commonly applied in works which calculate finite-horizon offline solutions for POMDP [70, 71]. The piecewise-linear, convex property of the optimal reward function has been shown in [70]. This model can allow this advantageous property to be exploited, to simplify the offline solution.

However, this thesis strives to solve the POMDP online, for which the reason will be explained in Section 4.3.1. The *belief states* of the agent will be applied to be the state of the problem, rather than the world state. Compared with the world state, which the agent can not access directly, *belief states* can more straightforwardly represent the knowledge of the agents about the environment, thus can facilitate the design of more intuitive strategy. The new POMDP formulation is as follows:

1. S is a finite set of belief states of the agent about the world. At time t, a belief state $s_t = \{\{\hat{\mathrm{P}}_\lambda(c,t|Y_t) : \lambda \in \Lambda\}, \{\hat{\mathrm{x}}_\lambda(t) : \lambda \in \Lambda_t\}, x_\rho(t), \Lambda_t, t\}$. $s_t \in S$;
2. A_s is an finite set of possible actions of the agent at state s. Let a denote the action of the agent, which is its movement between neighbouring cells. $a \in A_s$;
3. $p(s_{t+1}|s_t, a)$: $\boldsymbol{S} \times A_s \longrightarrow p(\boldsymbol{S})$ is the state-transition function, mapping from a previous belief state and an agent action, to a probability distribution of next belief states;
4. $R : S \times A \longrightarrow R$ is the reward function, mapping from a current belief state and an agent action to an immediate reward;
5. Ω is a finite set of observation of the pursuer;

6. $O : S \times A_s \longrightarrow p(\Omega)$ is the observation function, mapping from a current belief state and an agent action, to a probability distribution of agent observations.

where ρ is the ID of the only pursuer. The state transition function $p(s_{t+1}|s_t, a)$ is based on the update rules of $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$, $\hat{\mathrm{x}}_\lambda(t)$, and Λ_t, which are defined in Chapter 3.

4.2 Construction of the Objective Function

In Section 3.4.2.2, the *belief probability* is defined to represent the certainty about a target, which the pursuer should increase. Let $\tilde{B}_\lambda(s_t)$ be the *belief probability* of target λ, at state s_t. Let $R(s_t) = \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(s_t)$ be the reward for SSM mission. It provides the lower bound for the expected number of targets which can be detected, if at time t, $m = |\Lambda_t|$ agents are deployed to reach the estimated location of each known target.

π denotes the policy of the agent, to decide which action to take at each time step. s_{t_f} denotes one of the possible terminal states, and $p(s_t|\pi, s_{t_i})$ is its probability distribution on S, given pursuit policy π and initial state s_{t_i}. According to [95], the objective function for SSM, for time horizon $T = t_f - t_i$, is formulated as the expected average of the rewards for all time steps within time horizon:

$$
\begin{aligned}
V(\pi, s_{t_i}, t_f) &= E\{\frac{\Delta T}{T} \sum_{t=t_i}^{t_f} R(s_t)\} = \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} E\{R(s_t)\} \\
&= \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\pi, s_{t_i}) R(s_t) \\
&= \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\pi, s_{t_i}) \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(s_t)
\end{aligned}
$$

where ΔT is the time step of system.

In Section 3.4.2.2, it has been introduced that, the expected *belief probability* of a known target can be increased by having an agent to traverse its estimated location, and the total *belief probability* can be increased by searching unknown targets and monitoring known ones at the same time. Then, the agent needs to plan a strategy to do search and monitoring simultaneously and cooperatively, in order to achieve a higher objective value.

4.3 Policy Planning for SSM

Through Section 4.1 and 4.2, a tuple $\langle S, A_s, p(s_{t+1}|s_t, a), V(\pi, s_{t_i}, t_f)\rangle$ has been defined for the single pursuer SSM mission, which is a full model of POMDP. The belief state s_t represents the real time knowledge of the pursuer about the environment, and the transition function $p(s_{t+1}|s_t, a)$ determines the evolution of the environment. The objective function $V(\pi, s_{t_i}, t_f)$ builds up the goals for the SSM mission. The solution to this POMDP is the policy π of the agent, which is a strategy that determines the actions of the pursuer, choosing from the action pool A_s.

4.3.1 Concept for Solving POMDP

Although the problem in this chapter is built as a POMDP, the belief state rather than the world state is taken as the state of the system, which can be accessed directly by the agent. Hence this POMDP can be viewed a subjective MDP [68]. Lemma 4.1 proves the existence of a solution to this subjective MDP.

Lemma 4.1. *For the Finite-Horizon subjective MDP defined by tuple* $\langle S, A_s, p(s_{t+1}|s_t, a)..$ $, V(\pi, s_{t_i}, t_f)\rangle$*, there exists a deterministic history dependent policy* $a_t = \pi^*(s_t, h_t)$*, which can achieve the optimal objective value. Where* $h_t = (h_{t-1}, a_{t-1}, s_t)$ *denotes system history. [95]*

The definition of the optimal policy is explained in Figure 4.1. The decision tree roots from the initial state and ends at a certain horizon. Each action on the initial state may result in several possible states, so do the subsequent actions. Each policy is a subtree of the decision tree, which decides what actions to take at each state and each step. The optimal policy $\pi^*(s_t, h_t) = argmax_\pi V(\pi, s_{t_i}, t_f)$ is the subtree (Green) that can achieve the highest objective value.

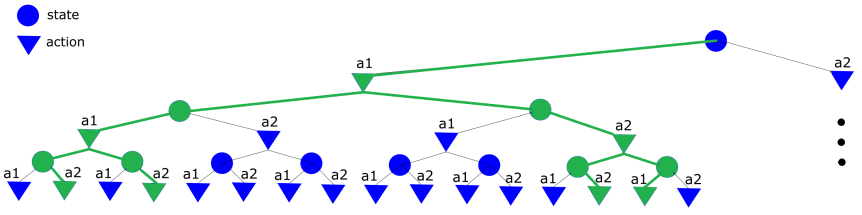


FIGURE 4.1: optimal policy

The solution methods for POMDP has been reviewed in Section 2.2.1, which in general can be categorized as offline solutions and online solutions. The offline solutions plan a comprehensive strategy before the mission, and implement it without replanning.

Although being fast to execute, the offline strategy can be very slow to plan because of the exhaustive backup. The computational cost of offline solution grows significantly with the size of the problem. Thus it normally does not allow real-time replanning of the strategy.

However, from Chapter 3, it can be seen that the SSM problem contains $m+n$ players and an area with $n_x \times n_y$ grid cells. The internal MDP, which does not include the partial observability of the agents, already has a state space with a size of $(n_x \times n_y)^{m+n}$. The complexity of a comprehensive offline solution should grow at least exponentially with the number of the players, and also polynomially with the size of the area. Hence the offline methods are not scalable with the size of this problem. Besides, the SSM is a information gathering problem, which may deal with uncertainties on the model of environment, the model of players, and the number of targets. These uncertainties require the agent to be capable of replanning its policy, to adapt to the changes. Apparently, the offline solution does not satisfy this requirement.

An alternative is online planning method. Based on current state, it only explores the part of state space which are reachable within a time horizon from the initial state, then plans a local policy which considers the explored subset of state space. The local policy will be implemented until reaching the time horizon or the occurrence of certain events, and will then be replanned. This approach focuses on local information and the close future within time horizon, thus it has a deterministic upper bound for the size of the decision tree, and can be computed in real time with moderate cost. It can easily take into account the environmental changes in each real-time planning, to make it adaptive [81]. Therefore, the online planning is chosen in this thesis to solve the POMDP, to have a more feasible, scalable, and flexible solution.

The online solution of POMDP can mainly be either dynamic programming or policy iteration. The dynamic programming method is based on solving the Bellman Equation, where

$$\begin{aligned} &V(\pi, s_{t_i}, t_f) = \\ &\frac{\Delta T}{T} R(s_{t_i}) + \frac{T - \Delta T}{T} \sum_{s_{t_i+1} \in S} p(s_{t_i+1} | s_{t_i}, a) V(\pi, s_{t_i+1}, t_f) \end{aligned} \tag{4.1}$$

It can be seen that, Dynamic Programming method searches in the state space, and works out the optimal policy. Instead, policy iteration searches in the policy space directly [78, 96, 97]. The concept of policy iteration is chosen in this thesis, to be the solution for the online policy planning. Although policy iteration is not simpler

than Dynamic Programming in terms of getting an optimal solution, but it allows the heuristics formulation of the policy space, which enables utilizing the physical properties of the SSM to simplify the planning. This advantage will be exploited from Section 4.3.2 to 4.3.3.4

In any method, solving POMDP precisely can be difficult [72]. To have a feasible online planning, a sub-optimal solution is chosen, which strives to find a close approximation of the optimal policy π^*. The basic framework for solving POMDP in this thesis is as follows:

1. Build a structure of policy $\hat{\pi}$, which has the potential to approximate the optimal policy π^*.
2. Find the $\hat{\pi}^* = argmax_{\hat{\pi}} V(\hat{\pi}, s_{t_i}, t_f)$, which is the best policy to be obtained within the structure of $\hat{\pi}$.

Subsection 4.3.2 to 4.3.3.4 detail how to build the structure of $\hat{\pi}$. Subsection 4.3.4 and Section 4.4 describe how to calculate $\hat{\pi}^*$.

4.3.2 Simplifications for State Prediction

When building the search tree illustrated in Figure 4.1, the agent needs to predict the future states, which are the nodes in the search tree. It is a main part of computation. Therefore, to reduce computational complexity, the following assumptions/simplifications are made for the state prediction in planning.

1. **Perfect Sensor Assumption**. Assume that for the sensor model in equation (3.1), $p(1|0) = 0, p(0|1) = 0$.
2. **Contingency Density Assumption**. Assume that at each time step, only one contingency may happen. The contingencies can be four kinds of events: 1. detecting a new target, 2. re-detecting a known target, 3. losing a known target, 4. other events, where there is no detection or loss of a target.
3. **Probability Distribution Update Simplification**. $\hat{\mathrm{P}}_u(c, t|Y_t)$ is estimated by both target dynamics and sensing. In policy planning, for a future time instant, the influence of both target behaviour and sensing on $\hat{\mathrm{P}}_u(c, t|Y_t)$ is ignored.
4. **Location Update Assumption**. Once a known target λ is redetected, $\hat{\mathrm{x}}_\lambda(t)$ will be updated. Assume that this update does not dramatically change s_t.

Assumption 1 is valid if it is confident enough about the accuracy of target detection sensor. The sensing errors during planning will not be considered. Although it will be tested during simulation.

Assumption 2) is based on the fact that the environment is large and the targets are assumed to be sparsely scattered. The case when different events happen at the same time is ignored. Based on assumption 2), the states in which the agent has no detection or loss of a target are classified as s' (with event 4). The other states are classified as s°, in which the agent detects an unknown target, re-detect or lose a known target (with event 1,2, or 3). States s° are called branching states.

Based on assumptions 3), a simplification is made that, when predicting the future probability distribution of targets, $\hat{\mathrm{P}}_u(c,t|Y_t)$ is viewed as a fixed, which is $\hat{\mathrm{P}}_u(c,t|Y_t) = \hat{\mathrm{P}}_u(c,t_i|Y_{t_i})$. The induced error is compensated by not planing an agent to search for a location twice within time horizon, to avoid including the chance of detection repetitively for an area. According to 4), when predicting the future estimated location of targets, the update of $\hat{\mathrm{x}}_\lambda(t)$ is ignored, which means $\hat{\mathrm{x}}_\lambda(t) = \hat{\mathrm{x}}_\lambda(t_i)$.

It should be noted that all these simplifications only apply to the phrase of planning, when the agent needs to predict future state. It does not apply to the estimation of current information during the execution of a policy.

4.3.3 Policy Reconstruction

As shown in Section 4.3.1, it is not practical to solve the POMDP by doing an exhaustive enumeration. Thus an approximation method is needed to obtain an estimation of the optimal policy and the future objective value. The conventional decision tree shown in Figure 4.1 treats all the branching with no preference of importance, thus is not efficient and is not convenient to incorporate heuristics. Therefore, a novel policy reconstruction concept is proposed, to make room for decomposing a policy and having intuitive approximation.

At initial state s_{t_i}, a deterministic trajectory is proposed for the agent: $\chi = \{x'(t) : t = t_i, t_i+1, ..., t_f, x'(t_i) = x_\rho(t_i)\}$, which is called base trajectory. The base trajectory always starts from the location of the pursuer. $x'(t)$ denotes the location that the agent is planning to visit at time t. The base trajectory includes a set of target locations $X(t_i) = \{\hat{\mathrm{x}}_\lambda(t_i) : \lambda \in H(t_i)\}$, where $H(t_i) \in \Lambda(t_i)$ is the set of known targets to be monitored along χ.

Assume that there is a branching function $\chi^\circ = f(s^\circ, \chi, h_t)$ which maps a branching state s°, current base trajectory χ and system history h_t, to a new base trajectory χ°.

The beginning of the new base trajectory is the location of pursuer when the branching happens. Also let $\{a_t : a_t = x_\rho(t+1), t \in \{t_i, t_i+1, ..., t_f-1\}\}$ be the action taken by the agent to decide the next immediate location. A policy structure $\hat{\pi}$ is defined in Algorithm 1:

Algorithm 1: $a_t = \hat{\pi}(s_t, \chi, h_t)$

if $s_t \in s^\circ$ **then**
  $\chi^\circ = f(s^\circ, \chi, h_t)$
  $\chi = \chi^\circ$
end
$a_t = x_\rho(t+1) \in \chi$

where $h_t = (h_{t-1}, a_{t-1}, s_t)$ denotes system history.

The base trajectory can be viewed as a provisional plan of the agent trying to explore certain unknown area and visit some known targets. The branching function is a reaction scheme to amend the previous plan. Then it will be proved that the combination of a base trajectory and a branching function can be equivalent to the decision tree description of a policy.

Theorem 4.2. *For the POMDP defined by tuple* $(S, A_s, p(s_{t+1}|s_t, a), V(\pi, s_{t_i}, t_f))$*, there exists a deterministic history dependent policy* $\hat{\pi}^*(s_t, \chi, h_t)$ *defined in Algorithm 1, to be optimal.*

Proof. Assume that there is an arbitrary deterministic history dependent policy $\pi(s_t, h_t)$, applied on an arbitrary initial state s_{t_i}, with the terminal time at t_f. Let $a_{t_i} = \pi(s_{t_i}, h_{t_i})$ be the first action. For all the later states, if $\{s_t : t = t_i+1, ...t_f\} \in s'$, let $\{a'_t : t = t_i+1, ...t_f, a'_t = \pi(s_t, h_t)\}$ denotes the corresponding sequence of actions taken by the policy.

Because the category of states s' corresponds to a deterministic kind of observation, and plus the policy $\pi(s_t, h_t)$ is know, then the sequences of the states $\{s_t : t = t_i+1, ...t_f\} \in s'$ and the actions $\{a'_t : t = t_i+1, ...t_f, a'_t = \pi(s_t, h_t)\}$ are deterministic. Then, there should be a deterministic trajectory $\chi = \{a_{t_i}, a'_{t_i+1}, ..., a'_{t_f}\}$.

If at time t_1, $s_{t_1} \in s^\circ$, then the immediate action taken is $a^\circ_{t_1} = \pi(s_{t_1}, h_{t_1})$, and let $\{a''_t : t = t_{1+}, ...t_f, a''_t = \pi(s_t, h_t)\}$ denote all the corresponding actions for later non-branching states $s_t \in s'$. Let $\chi^\circ = \{a^\circ_{t_1}, a''_{t_{1+}}, ..., a''_{t_f}\}$.

It can be seen that after iteratively applying this process, all possible states and the corresponding actions in the policy tree can be reconstructed by the combination of χ and all branching χ°, which means that $\pi(s_t, h_t)$ can be fully reconstructed by $\hat{\pi}(s_t, \chi, h_t)$

defined in Algorithm 1. According to Lemma 4.1, there exist an optimal $\pi^*(s_t, h_t)$. Thus it can be reconstructed by a $\hat{\pi}^*(s_t, \chi, h_t)$, which is also optimal. □

Theorem 4.2 and its proof show that the optimal policy π^* can be exactly reconstructed by $\hat{\pi}^*(s_t, \chi, h_t)$. It can be also illustrated in Figure 4.2 that any policy, such as the optimal policy in Figure 4.1, can be reconstructed by such set-up, where a different colour of branching denote a base trajectory.

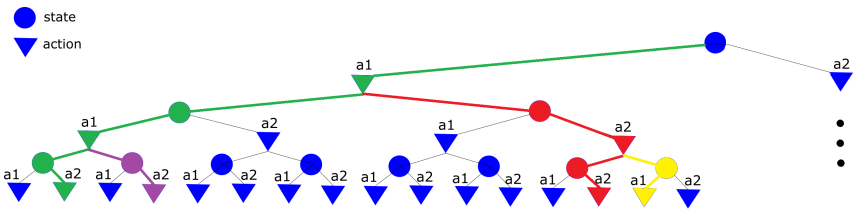


FIGURE 4.2: policy reconstruction

Such decomposing and reconstruction do not simplify the policy planning, but both the base trajectory and the branching function can be interpreted in an intuitive way, which makes it easy to do approximation with heuristics and domain knowledge. Then, with such formulation, a reconstructed structure of the agent policy can be built, which incorporates heuristics to imitate the sensible actions taken by the optimal policy, and have the potential to approximate the optimal policy and be sub-optimal.

4.3.3.1 Fixed Sequence of Actions vs. Reactive Policy

In [77, 80, 98, 99], and [81], different approaches for approximating the optimal policy are introduced. Amongst those approaches, the strategy of fixed sequence of actions is commonly used in relevant problems [59], for its ease of implementation. It plans a policy $a_t = \pi(t, h_t)$, which is a fixed sequence of actions within the time horizon, regardless of the possible future states. In the problem of this work, such fixed sequence of actions is a deterministic path for the agent. It has been stated in [86] that such solution can guarantee a lower bound of optimal reward. In [98], it is proven that such optimization can be at least as well as an optimal open-loop policy, which means the current information is advantageously utilized in the planning. The fixed sequence of action method is illustrated in Figure 4.3.

It can be seen from Figure 4.3 that, in such policy, the action at each time step is fixed, regardless any branchings. Comparing Figure 4.3 and 4.1, we can see that it is almost impossible for a fixed sequence of actions to precisely approximate the optimal policy $a_t = \pi^*(s_t, h_t)$. In this work, as mentioned in Chapter 1, the state and objective value

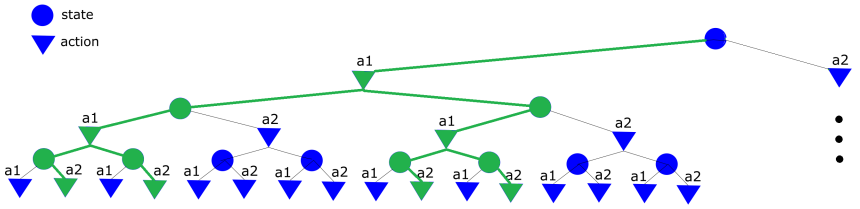


FIGURE 4.3: fixed sequence of actions

may be largely changed by contingencies such as a new detection or a failed monitoring, thus the branchings triggered by these events should be considered in the planning of a strategy. The obtained strategy should be reactive to future contingencies.

In Section 4.3.3.2, 4.3.3.3, and 4.3.3.4, three different structures of policies will be proposed, which include a policy with fixed sequence of actions, a hybrid policy of fixed sequence of actions and branching, and a heuristic reactive policy. These policies will be compared theoretically and through simulation.

4.3.3.2 Policy of Fixed Sequence of Actions

For the decomposing of agent policy defined in Section 4.3.3, one most simple way to do approximation is to ignore the branching function. Then the base trajectory will not change regardless of any events happening.

Let χ to be a base trajectory for the agent to follow. For any branching state s° to happen, no change to χ will be applied, thus the Algorithm 1 becomes as follows:

Algorithm 2: $a_t = \hat{\pi}_f(s_t, \chi, h_t)$

if $s_t \in s^\circ$ **then**
  $\chi^\circ = \chi$
  $\chi = \chi^\circ$
end
$a_t = x_\rho(t+1) \in \chi$

This is called a policy of fixed sequence of actions (FSOA). For such policy, the action taken by the agent at each time step is deterministic. Based on the belief state transition function $p(s_{t+1}|s_t, a)$, which is known, the evolution of the probability distribution of the belief state, $p(s_t|\hat{\pi}_f, s_{t_i})$, is deterministic as well. Then the objective value $V(\hat{\pi}_f, s_{t_i}, t_f) = \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\hat{\pi}_f, s_{t_i}) R(s_t)$ can be calculated straightforwardly. This property will be utilized in Section 4.3.4, to calculate the objective value of other policies which will be proposed.

Let $\hat{\pi}_f^* = argmax_{\hat{\pi}_f} V(\hat{\pi}_f(s_t, \chi, h_t), s_{t_i}, t_f)$ to be the optimal policy with fixed sequence of actions. Let $\hat{\pi}_o^* = argmax_{\hat{\pi}_f} \sum_{s_{t_i} \in S} p(s_{t_i}|s_{t_i-1}, a(t_i-1)) V(\hat{\pi}_f(s_t, \chi, h_t), s_{t_i}, t_f)$ to be the optimal open-loop policy, which is still a fixed sequence of actions but do not consider the current observation.

Lemma 4.3. *For a optimal policy of FSOA, which is $\hat{\pi}_f^* = argmax_{\hat{\pi}_f} V(\hat{\pi}_f(s_t, \chi, h_t), s_{t_i}, t_f)$, its estimated objective value is a lower bound of the objective value achieved by optimal policy π^* [86].*

Lemma 4.4. *For an optimal policy of FSOA, which is $\hat{\pi}_f^* = argmax_{\hat{\pi}_f} V(\hat{\pi}_f(s_t, \chi, h_t), s_{t_i}, t_f)$, its estimated objective value is at least higher than that of the optimal open-loop policy $\hat{\pi}_o^*$ [98].*

Both Lemmas provide guarantee for the performance of policy of fixed sequence of actions.

4.3.3.3 Hybrid Policy of Fixed Sequence of Actions and Branching

To address the drawback of the policy of FSOA, as mentioned in Section 4.3.3.1, it is mixed with a fully reactive policy. The mixed policy, $\hat{\pi}_m(s_t, \chi, h_t)$, is called hybrid policy of fixed sequence of actions and branching, which will be mentioned as hybrid policy in the rest of this thesis. Assume that in policy $\hat{\pi}_m(s_t, \chi, h_t)$, the agent can do K levels of branchings to change the base trajectory. The base trajectory remain unchanged after the Kth branching. Such policy maintains a certain depth of branching, but still uses fixed sequence of actions to approximate the decisions further than that depth, thus is a trade-off between optimality and computational efficiency. This policy is defined in Algorithm 3.

Algorithm 3: $a_t = \hat{\pi}_m(s_t, \chi, h_t)$

initialize i=0
if $s_t \in s^\circ$ **then**
  $i = i + 1$
  $\chi^\circ = f_m(s^\circ, \chi, h_t)$
  $\chi = \chi^\circ$
end
$a_t = x_\rho(t+1) \in \chi$

where $f_m(s_t, \chi, h_t) = \begin{cases} argmax_\chi V(\hat{\pi}_m(s_t, \chi, h_t), s_t, t_f) & \text{if } i \leq K \\ \chi & \text{if } i > K \end{cases}$

It can be seen that by limiting the maximum levels of branchings to happen, the hybrid policy does trade-off between computational efficiency and the look-ahead ability of future contingencies. The definition of the branching function $f_m(s^\circ, \chi, h_t)$ contains

finding the best branching with the highest objective value, which is similar to solving the Bellman Equation, and is as difficult. But the policy after the Kth branching will be a FSOA. As mentioned in Section 4.3.3.2, the objective value of such non-reactive policy can be computed conveniently. Then the objective value of ith branching can be calculated recursively, inducing backwards from the Kth branching. The branching function $f_m(s^\circ, \chi, h_t)$ can be determined in such a way, and so is the hybrid policy $\hat{\pi}_m$. $\hat{\pi}_m^* = argmax_{\hat{\pi}_m} V(\hat{\pi}_m(s_t, \chi, h_t), s_{t_i}, t_f)$ is the optimal hybrid policy of fixed sequence of actions and branching. It has the following property:

Theorem 4.5. *The optimal hybrid policy $\hat{\pi}_m^*$ has at least higher estimated objective value than that of optimal policy of FSOA $\hat{\pi}_f^*$.*

Proof. Assume that there is an optimal policy of FSOA $a_t = \hat{\pi}_f^*(s_t, \chi_f, h_t)$. A hybrid policy $a_t = \hat{\pi}_m(s_t, \chi_f, h_t)$ is built based on Algorithm 3, which takes χ_f as the initial trajectory and has at most K level of branchings.

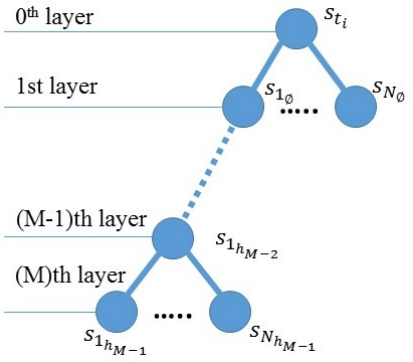


FIGURE 4.4: branching tree

For an agent applying policy $\hat{\pi}_m$, all the branchings are triggered by branching state $s \in s^\circ$. Assume that, for the agent to follow a base trajectory χ, there may be several possible branching events to happen through χ. x denotes the number in a sequence, of a branching event along χ. $x_{h_{k-1}}$ denotes the xth branching event, given $h_{k-1} = \{\chi_{k-1}, x'_{h_{k-2}}, h_{k-2}\}$ which is the history of prior branchings and base trajectories. For the history h_{k-1}, $x'_{h_{k-2}}$ is the previous branching event happened before $x_{h_{k-1}}$, and $\chi_{k-1} = \chi^\circ_{x'_{h_{k-2}}}$ is the base trajectory adjusted after the occurrence of $x'_{h_{k-2}}$, which is also the base trajectory where $x_{h_{k-1}}$ happens along. $s_{x_{h_{k-1}}}$ denotes the state that the $x_{h_{k-1}}$ happens. $h_0 = \phi$ denotes that there is no priori branching at the root of the decision tree. Let $N_{h_{k-1}}$ denote the maximum number of possible branchings along $\chi_{k-1} \in h_{k-1}$ given history h_{k-1}. There can only be at most $K \leq T/\Delta T$ levels of branchings within time horizon, which is defined in policy $\hat{\pi}_m$. The branching tree is

illustrated in Figure 4.4, where $M_h \leq K$ is the maximum levels of branchings in current branch, given the history h.

Let $t_{x_{h_{k-1}}}$ and $p(x_{h_{k-1}})$ denote the time instant and probability of $x_{h_{k-1}}$ to occur, given the current base trajectory $\chi_{k-1} \in h_{k-1}$ and the history h_{k-1}. $p_{0_{h_{k-1}}}$ denotes the probability that no branching event happens along $\chi_{k-1} \in h_{k-1}$, given h_{k-1}. Let $V_h(t_1, t_2|h_{k-1})$ denote the hindsight objective value according to hindsight history from time t_1 to t_2, where no branching is made, given the current base trajectory $\chi_{k-1} \in h_{k-1}$ and the history h_{k-1}. Let $V_f(\chi_f, s_{t_i}, t_f) = V(\hat{\pi}_f^*, s_{t_i}, t_f)$ to be the expected objective value of applying policy $\hat{\pi}_f^*$, and let $V_m(\chi_f, s_{t_i}, t_f) = V(\hat{\pi}_m, s_{t_i}, t_f)$ to be the objective value of applying hybrid policy $\hat{\pi}_m$. Thus $V_m(\chi_f, s_{t_i}, t_f)$ can be constructed as follows, considering all the possible branchings

$$\begin{aligned} V_m(\chi_f, s_{t_i}, t_f) &= p(0_\phi)V_h(t_i, t_f|\phi) + p(1_\phi)(\delta_{1_\phi}V_h(t_i, t_{1_\phi}|\phi) \\ &+ (1-\delta_{1_\phi})V_m(\chi^\circ_{1_\phi}, s_{1_\phi}, t_f)) + ... + p(N_\phi)(\delta_{N_\phi}V_h(t_i, t_{N_\phi}|\phi) \\ &+ (1-\delta_{N_\phi})V_m(\chi^\circ_{N_\phi}, s_{N_\phi}, t_f)); \end{aligned} \tag{4.2}$$

where $\delta_{x_{h_k}} = (t_{x_{h_k}} - t_{x'_{h_{k-1}}})/(t_f - t_{x'_{h_{k-1}}})$, $\chi^\circ_{x_{h_k}} = f_m(s^\circ_{x_{h_k}}, \chi^\circ_{x'_{h_{k-1}}}, h_k)$.

According to the definition of $f_m(s^\circ, \chi, h_t)$, $V_m(\chi^\circ_{x_\phi}, s_{x_\phi}, t_f) = argmax_\chi V_m(\chi, s_{x_\phi}, t_f) \geq V_m(\chi_f, s_{x_\phi}, t_f)$. Applying this property into Equation (4.2), it can be obtained that

$$\begin{aligned} V_m(\chi_f, s_{t_i}, t_f) &\geq p(0_\phi)V_h(t_i, t_f|\phi) + p(1_\phi)(\delta_{1_\phi}V_h(t_i, t_{1_\phi}|\phi) \\ &+ (1-\delta_{1_\phi})V_m(\chi_f, s_{1_\phi}, t_f)) + ... + p(N_\phi)(\delta_{N_\phi}V_h(t_i, t_{N_\phi}|\phi) \\ &+ (1-\delta_{N_\phi})V_m(\chi_f, s_{N_\phi}, t_f)); \end{aligned} \tag{4.3}$$

Applying the same property iteratively, we get

$$
\begin{aligned}
&V_m(\chi_f, s_{x_\phi}, t_f) = p(0_{h_1})V_h(t_{x_\phi}, t_f|h_1) + p(1_{h_1})(\delta_{1_{h_1}} V_h(t_i, t_{1_{x_\phi}}|h_1) \\
&+ (1-\delta_{1_{h_1}})V_m(\chi^\circ_{1_{h_1}}, s_{1_{h_1}}, t_f)) + ... + p(N_{h_1})(\delta_{N_{h_1}} V_h(t_{x_\phi}, t_{N_{h_1}}|h_1) \\
&+ (1-\delta_{N_{h_1}})V_m(\chi^\circ_{N_{h_1}}, s_{N_{h_1}}, t_f)) \\
&\geq p(0_{h_1})V_h(t_{x_\phi}, t_f|h_1) + p(1_{h_1})(\delta_{1_{h_1}} V_h(t_i, t_{1_{x_\phi}}|h_1) \\
&+ (1-\delta_{1_{h_1}})V_m(\chi_f, s_{1_{h_1}}, t_f)) + ... + p(N_{h_1})(\delta_{N_{h_1}} V_h(t_{x_\phi}, t_{N_{h_1}}|h_1) \\
&+ (1-\delta_{N_{h_1}})V_m(\chi_f, s_{N_{h_1}}, t_f)); \\
&... \\
&V_m(\chi_f, s_{x_{h_{k-1}}}, t_f) = p(0_{h_k})V_h(t_{x_{h_{k-1}}}, t_f|h_k) + p(1_{h_k})(\delta_{1_{h_k}} \\
&V_h(t_{x_{h_{k-1}}}, t_{1_{h_k}}|h_k) + (1-\delta_{1_{h_k}})V_m(\chi^\circ_{1_{h_k}}, s_{1_{h_k}}, t_f)) + ... + p(N_{h_k}) \\
&(\delta_{N_{h_k}} V_h(t_{x_{h_{k-1}}}, t_{N_{h_k}}|h_k) + (1-\delta_{N_{h_k}})V_m(\chi^\circ_{N_{h_k}}, s_{N_{h_k}}, t_f)) \\
&\geq p(0_{h_k})V_h(t_{x_{h_{k-1}}}, t_f|h_k) + p(1_{h_k})(\delta_{1_{h_k}} \\
&V_h(t_{x_{h_{k-1}}}, t_{1_{h_k}}|h_k) + (1-\delta_{1_{h_k}})V_m(\chi_f, s_{1_{h_k}}, t_f)) + ... + p(N_{h_k}) \\
&(\delta_{N_{h_k}} V_h(t_{x_{h_{k-1}}}, t_{N_{h_k}}|h_k) + (1-\delta_{N_{h_k}})V_m(\chi_f, s_{N_{h_k}}, t_f)); \\
&...k \in [1, M_h - 1]
\end{aligned}
\tag{4.4}
$$

where $M_h \leq K$ is the maximum levels of branchings in current branch, given the history h.

Because there will be no more branchings after $x_{h_{M_h-1}}$, and based on the definition of $f_m(s^\circ, \chi, h_t)$, $V_m(\chi_f, s_{x_{h_{M_h-1}}}, t_f) = V_f(\chi_f, s_{x_{h_{M_h-1}}}, t_f)$. Apply this result to Equation (4.4), and it can be obtained that

$$
\begin{aligned}
&V_m(\chi_f, s_{x_{h_{M_h-2}}}, t_f) = p(0_{h_{M_h-1}}) V_h(t_{x_{h_{M_h-2}}}, t_f | h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}} V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}} | h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}}) V_m(\chi^{\circ}_{1_{h_{M_h-1}}}, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}} V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}} | h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V_m(\chi^{\circ}_{N_{h_{M_h-1}}}, s_{N_{h_{M_h-1}}}, t_f)) \\
&\geq p(0_{h_{M_h-1}}) V_h(t_{x_{h_{M_h-2}}}, t_f | h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}} V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}} | h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}}) V_m(\chi_f, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}} V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}} | h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V_m(\chi_f, s_{N_{h_{M_h-1}}}, t_f)) \\
&= p(0_{h_{M_h-1}}) V_h(t_{x_{h_{M_h-2}}}, t_f | h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}} V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}} | h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}}) V_f(\chi_f, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}} V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}} | h_{M_h-1}) \\
&+ (1-\delta_{N_{h_{M_h-1}}}) V_f(\chi_f, s_{N_{h_{M_h-1}}}, t_f)) \\
&= V_f(\chi_f, s_{x_{h_{M_h-2}}}, t_f)
\end{aligned}
\tag{4.5}
$$

Thus we get that $V_m(\chi_f, s_{x_{h_{M_h-2}}}, t_f) \geq V_f(\chi_f, s_{x_{h_{M_h-2}}}, t_f)$. Applying this property and the same process in Equation (4.4) iteratively, it can be seen that

$$
\begin{aligned}
&V_m(\chi_f, s_{x_{h_{k-1}}}, t_f) \geq V_f(\chi_f, s_{x_{h_{k-1}}}, t_f) \\
&... \\
&V_m(\chi_f, s_{t_i}, t_f) \geq V_f(\chi_f, s_{t_i}, t_f)
\end{aligned}
$$

Thus it proves that given an optimal policy of FSOA $\hat{\pi}^*_f(s_t, \chi_f, h_t)$, there will always be a hybrid policy $\hat{\pi}_m(s_t, \chi_f, h_t)$ to achieve at least better estimated objective value, which proves the theorem. □

Theorem 4.5 proves that, combining branching and FSOA can improve the performance of the policy of FSOA.

Theorem 4.6. *When the maximum level of branching for the optimal hybrid policy $\hat{\pi}^*_m$ is equal to or bigger than the depth of the search tree, which is when $K \geq T/\Delta T$, the optimal hybrid policy $\hat{\pi}^*_m$ is equivalent to the optimal policy π^*.*

Proof. According to Theorem 4.2, there is a policy $\hat{\pi}^*(s_t, \chi^*, h_t)$ defined in Algorithm 1, to be equivalent to the optimal policy π^*. Let $K \geq T/\Delta T$, which is the number of time

steps within the time horizon, and is also the maximum number of levels of branching. Let $a_t = \hat{\pi}^*_m(s_t, \chi^*, h_t)$ be the hybrid policy which take χ^* as the base trajectory. Using the same notation in the proof of Theorem 4.5, let $V(\hat{\pi}^*, s_{t_i}, t_f)$ be the optimal expected objective value of applying optimal policy $\hat{\pi}^*$, and let $V_m(\chi^*, s_{t_i}, t_f) = V(\hat{\pi}^*_m, s_{t_i}, t_f)$ be the objective value of applying optimal hybrid policy $\hat{\pi}^*_m$. Then $V_m(\chi^*, s_{t_i}, t_f)$ can be constructed as follows, considering all the possible branchings

$$
\begin{aligned}
&V_m(\chi^*, s_{t_i}, t_f) = p(0_\phi)V_h(t_i, t_f|\phi) + p(1_\phi)(\delta_{1_\phi}V_h(t_i, t_{1_\phi}|\phi) \\
&+ (1-\delta_{1_\phi})V_m(\chi^\circ_{1_\phi}, s_{1_\phi}, t_f)) + ... + p(N_\phi)(\delta_{N_\phi}V_h(t_i, t_{N_\phi}|\phi) \\
&+ (1-\delta_{N_\phi})V_m(\chi^\circ_{N_\phi}, s_{N_\phi}, t_f)); \\
&... \\
&V_m(\chi^\circ_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) = p(0_{h_k})V_h(t_{x_{h_{k-1}}}, t_f|h_k) + p(1_{h_k})(\delta_{1_{h_k}} \\
&V_h(t_{x_{h_{k-1}}}, t_{1_{h_k}}|h_k) + (1-\delta_{1_{h_k}})V_m(\chi^\circ_{1_{h_k}}, s_{1_{h_k}}, t_f)) + ... + p(N_{h_k}) \\
&(\delta_{N_{h_k}}V_h(t_{x_{h_{k-1}}}, t_{N_{h_k}}|h_k) + (1-\delta_{N_{h_k}})V_m(\chi^\circ_{N_{h_k}}, s_{N_{h_k}}, t_f)); \\
&...k \in [1, M_h - 1]
\end{aligned}
\tag{4.6}
$$

where $M_h \leq T/\Delta T \leq K$ is the maximum levels of branchings in current branch, given the history h. $\delta_{x_{h_k}} = (t_{x_{h_k}} - t_{x'_{h_{k-1}}})/(t_f - t_{x'_{h_{k-1}}})$, $\chi^\circ_{x_{h_k}} = f_m(s^\circ_{x_{h_k}}, \chi^\circ_{x'_{h_{k-1}}}, h_k)$.

Because there is no possible branching after $x_{h_{M_h-1}}$. Then $Vm(\chi^\circ_{x_{h_{M-1}}}, s_{x_{h_{M-1}}}, t_f)...$ $= V(\hat{\pi}^*, s_{x_{h_{M-1}}}, t_f)$. Therefore, we obtain that

$$
\begin{aligned}
&V_m(\chi^\circ_{x_{h_{M-2}}}, s_{x_{h_{M_h-2}}}, t_f) = p(0_{h_{M_h-1}})V_h(t_{x_{h_{M_h-2}}}, t_f|h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}})V_m(\chi^\circ_{1_{h_{M_h-1}}}, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V_m(\chi^\circ_{N_{h_{M_h-1}}}, s_{N_{h_{M_h-1}}}, t_f)) \\
&= p(0_{h_{M_h-1}})V_h(t_{x_{h_{M_h-2}}}, t_f|h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}})V(\hat{\pi}^*, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V(\hat{\pi}^*, s_{N_{h_{M_h-1}}}, t_f))
\end{aligned}
\tag{4.7}
$$

Based on the definition of $f_m(s^\circ, \chi, h_t)$, $V_m(\chi^\circ_{x_{h_{M-2}}}, s_{x_{h_{M_h-2}}}, t_f) = \max_\chi V_m(\chi, s_{x_{h_{M_h-2}}}, t_f)$. Thus,

$$
\begin{aligned}
&V_m(\chi^{\circ}_{x_{h_{M-2}}}, s_{x_{h_{M_h-2}}}, t_f) = \max_{\chi}(p(0_{h_{M_h-1}})V_h(t_{x_{h_{M_h-2}}}, t_f|h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}})V_m(\chi^{\circ}_{1_{h_{M_h-1}}}, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V_m(\chi^{\circ}_{N_{h_{M_h-1}}}, s_{N_{h_{M_h-1}}}, t_f))) \\
&= \max_{\chi}(p(0_{h_{M_h-1}})V_h(t_{x_{h_{M_h-2}}}, t_f|h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}})V(\hat{\pi}^*, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V(\hat{\pi}^*, s_{N_{h_{M_h-1}}}, t_f))) \\
&= V(\hat{\pi}^*, s_{N_{h_{M_h-2}}}, t_f))
\end{aligned}
\tag{4.8}
$$

Applying the same process iteratively, it can be seen that

$$
\begin{aligned}
&V_m(\chi^{\circ}_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) = V(\hat{\pi}^*, s_{x_{h_{k-1}}}, t_f) \\
&... \\
&Vm(\chi^*, s_{t_i}, t_f) = V(\hat{\pi}^*, s_{t_i}, t_f)
\end{aligned}
$$

Thus we see that the hybrid policy which takes χ^* as the base trajectory, can have the same objective value with the optimal policy. Thus it is proven that the policy $\hat{\pi}^*_m(s_t, \chi^*, h_t)$ is equivalent to the optimal policy π^*.

□

Theorem 4.6 shows that, with enough depth of branching, the hybrid policy has the potential to precisely reconstruct the optimal policy. Combined with the sub-optimality of hybrid policy shown in Theorem 4.5, we can see that, the hybrid policy is a trade-off between the optimality of the optimal policy π^* and the computational efficiency of the policy of FSOA. A bigger depth of branching K prefers optimality, and the lower K prefers efficiency.

4.3.3.4 Heuristic Reactive Policy

The hybrid policy is a rigorous method, with proven sub-optimality and approximation to the optimal policy. However, with the increase of the depth of branching, the computational complexity still grows exponentially. Thus to have a practical real time computation, the depth of branching should be limited to be much lower than the number of time steps within time horizon. Then, without considering the branching in further steps, the look-ahead ability of future contingencies and reactions, which should be a vital ability for the performance in a highly unpredictable and varying mission environment, is still constrained.

An alternative way is to include heuristics into the branching function. If there is a heuristic branching function to approximate the sensible reaction to contingencies, a forward induction in the policy planning can be done, which is straightforward and efficient in calculation. It is less rigorous than the hybrid policy, but it has a better look-ahead ability in planning, by being able to consider all the possible branchings. Its sub-optimality compared with fixed sequence of actions will be proven.

For the design of branching function, it takes advantage of the decomposition of policy introduced in Section 4.3.3. Some heuristics are incorporated into the branching function $f(s^\circ, \chi, h_t)$ by applying specific knowledge about the SSM problem in this work. In a base trajectory χ which traverses target locations $X(t_i)$, the nodes to traverse known targets are called monitoring nodes. At states s', the agent will keep following χ. Thus only the reaction of $\chi^\circ = f_a(s^\circ, \chi, h_t)$ to the branching states s° is defined:

1. **Detecting a New Target, or Re-detecting a Known Target**. If there is a detection of a new target λ at time t_d, then $H(t_d) = H(t_d) \bigcup \lambda$. The remaining part of χ is χ^r. Let $f_a(s^\circ, \chi, h_t) = \chi^r$, which does not change the original path. If a known target is re-detected on the path of χ, let $f_a(s^\circ, \chi, h_t) = \chi^r$.

2. **Losing a Known Target**. If a known target λ is lost at time t_d, then $H(t_d) = H(t_d)/\lambda$, and the remaining part of χ is χ^r. Then χ^r is refined in three steps:

   (a) *Prune.* Remove all the monitoring nodes from χ^r which traverse target λ;

   (b) *Straighten.* For each pruned node, use a straight line to connect the possible monitoring nodes before and after the pruned one, to replace the original segments of path connecting between them. Thus χ^r is straightened to be χ^{rs};

   (c) *Complement.* The straightening may make χ^{rs} shorter than χ^r for a length of l_c. For the remaining monitoring nodes which are not pruned in all previous branchings, assume that there is a polyline P_l connecting them in their

original sequence. Truncate P_l to a length of l_c, and add it to the end of χ^{rs}, to obtain χ^{rsc}. When the initial χ is proposed at the start of planning, if its length is not enough to cover the whole planning horizon, *Complement* will also be applied to add it up for a full planning horizon.

$\hat{\pi}_f(s_t, \chi^{rsc}, h_t) = x_\rho(t+1) \in \chi^{rsc}$ and $\hat{\pi}_f(s_t, \chi^r, h_t) = x_\rho(t+1) \in \chi^r$ are two policies with FSOA with respect to χ^{rsc} and χ^r. The objective value, $V(\hat{\pi}_f(s_t, \chi^{rsc}, h_t), s_{t_d}, t_f)$ and $V(\hat{\pi}_f(s_t, \chi^r, h_t), s_{t_d}, t_f)$, can be calculated deterministically and straightforwardly as mentioned in Section 4.3.3.2. The rationale of $\hat{\pi}_s$ is to prune monitoring nodes of lost target, to focus on later search and monitoring; $\hat{\pi}_r$ maintain the old route on the contrary. Then two routes are chosen by doing a comparison:

$$\chi^c = \begin{cases} \chi^r & \text{if } V(\hat{\pi}_r, s_{t_d}, t_f) > V(\hat{\pi}_s, s_{t_d}, t_f) \\ \chi^{rsc} & \text{if } V(\hat{\pi}_s, s_{t_d}, t_f) > V(\hat{\pi}_r, s_{t_d}, t_f) \end{cases} \tag{4.9}$$

which is to compare and choose between two routes. Let $f_a(s^\circ, \chi, h_t) = \chi^c$.

The full $f_a(s^\circ, \chi, h_t)$ is presented in Algorithm 4.

Algorithm 4: $\chi^\circ = f_a(s^\circ, \chi, h_t)$

```
χ° = χ^r
if losing a known target then
    calculate χ^c based on χ°
    χ° = χ^c
end
output χ°
```

Let $a_t = \hat{\pi}_a(s_t, \chi, h_t)$ be a reactive policy which is of the structure defined in Algorithm 1, and contains branching function $f_a(s^\circ, \chi, h_t)$ defined in Algorithm 4. The rationale of this formulation of branching function is to react to the loss of a known target. The agent will compare the benefit of following the same base trajectory or pruning the location of the lost target to concentrate the resources on later exploration of unknown area and the monitoring of other known targets. For the branching function $f_a(s^\circ, \chi, h_t)$, the candidate new base trajectory χ^{rsc} and χ^r can be calculated conveniently, and $V(\hat{\pi}_r, s_{t_d}, t_f)$ and $V(\hat{\pi}_s, s_{t_d}, t_f)$ can be computed directly as well. Thus it can be said that the proposed heuristic reactive policy should be of higher computational efficiency compared with the hybrid policy. Only the reaction to the loss of a known target is considered, but it will later be proved that this policy will have better performance than the policy of FSOA.

Theorem 4.7. *The optimal heuristic reactive policy* $\hat{\pi}_a^* = argmax_{\hat{\pi}_a} V(\hat{\pi}_a(s_t, \chi, h_t), s_{t_i}, t_f)$ *has an better estimated objective value than that of optimal policy of FSOA* $\hat{\pi}_f^*$.

Proof. Assume that there is an optimal policy of fixed sequence of actions a_t $= \hat{\pi}_f^*(s_t, \chi_f, h_t) = x_\rho(t+1) \in \chi_f$. A heuristic reactive policy $a_t = \hat{\pi}_a(s_t, \chi_f, h_t)$ is built based on Algorithm 3, which takes χ_f as the initial trajectory. Using the same notation in the proof of Theorem 4.5, let $V_f(\chi_f, s_{t_i}, t_f) = V(\hat{\pi}_f^*, s_{t_i}, t_f)$ be the expected objective value of applying policy $\hat{\pi}_f^*$, and let $V_a(\chi_f, s_{t_i}, t_f) = V(\hat{\pi}_a, s_{t_i}, t_f)$ be the objective value of applying heuristic reactive policy $\hat{\pi}_a$. Thus $V_a(\chi_f, s_{t_i}, t_f)$ can be constructed as follows, including all the possible branchings

$$
\begin{aligned}
&V_a(\chi_f, s_{t_i}, t_f) = p(0_\phi)V_h(t_i, t_f|\phi) + p(1_\phi)(\delta_{1_\phi}V_h(t_i, t_{1_\phi}|\phi) \\
&+ (1-\delta_{1_\phi})V_a(\chi^\circ_{1_\phi}, s_{1_\phi}, t_f)) + ... + p(N_\phi)(\delta_{N_\phi}V_h(t_i, t_{N_\phi}|\phi) \\
&+ (1-\delta_{N_\phi})V_a(\chi^\circ_{N_\phi}, s_{N_\phi}, t_f)); \\
&... \\
&V_a(\chi_f, s_{x_\phi}, t_f) = p(0_{h_1})V_h(t_{x_\phi}, t_f|h_1) + p(1_{h_1})(\delta_{1_{h_1}}V_h(t_i, t_{1_{x_\phi}}|h_1) \\
&+ (1-\delta_{1_{h_1}})V_a(\chi^\circ_{1_{h_1}}, s_{1_{h_1}}, t_f)) + ... + p(N_{h_1})(\delta_{N_{h_1}}V_h(t_{x_\phi}, t_{N_{h_1}}|h_1) \\
&+ (1-\delta_{N_{h_1}})V_a(\chi^\circ_{N_{h_1}}, s_{N_{h_1}}, t_f)); \\
&... \\
&V_a(\chi_f, s_{x_{h_{k-1}}}, t_f) = p(0_{h_k})V_h(t_{x_{h_{k-1}}}, t_f|h_k) + p(1_{h_k})(\delta_{1_{h_k}} \\
&V_h(t_{x_{h_{k-1}}}, t_{1_{h_k}}|h_k) + (1-\delta_{1_{h_k}})V_a(\chi^\circ_{1_{h_k}}, s_{1_{h_k}}, t_f)) + ... + p(N_{h_k}) \\
&(\delta_{N_{h_k}}V_h(t_{x_{h_{k-1}}}, t_{N_{h_k}}|h_k) + (1-\delta_{N_{h_k}})V_a(\chi^\circ_{N_{h_k}}, s_{N_{h_k}}, t_f)); \\
&...k \in [1, M_h - 1]
\end{aligned}
\tag{4.10}
$$

where$M_h \leq T/\Delta T$ is the maximum levels of branchings in current branch, given the history h. $\delta_{x_{h_k}} = (t_{x_{h_k}} - t_{x'_{h_{k-1}}})/(t_f - t_{x'_{h_{k-1}}})$, $\chi^\circ_{x_{h_k}} = f_a(s^\circ_{x_{h_k}}, \chi^\circ_{x'_{h_{k-1}}}, h_k)$.

As there will be no more branching after $x_{h_{M_h-1}}$, then $V_a(\chi^\circ_{x_{h_{M_h-1}}}, s_{x_{h_{M_h-1}}}, t_f) = V_f(\chi^\circ_{x_{h_{M_h-1}}}, s_{x_{h_{M_h-1}}}, t_f)$. Based on the definition of $f_a(s^\circ, \chi, h_t)$, $V_a(\chi^\circ_{x_{h_{M_h-1}}}, s_{x_{h_{M_h-1}}}, t_f) = V_f(\chi^\circ_{x_{h_{M_h-1}}}, s_{x_{h_{M_h-1}}}, t_f) =\geq V_f(\chi^\circ_{x'_{h_{M_h-2}}}, s_{x_{h_{M_h-1}}}, t_f)$, where $x'_{h_{M_h-2}} \in h_{M_h-1}$, thus

$$
\begin{aligned}
&V_a(\chi^{\circ}_{x_{h_{M_h-2}}}, s_{x_{h_{M_h-2}}}, t_f) = p(0_{h_{M_h-1}})V_h(t_{x_{h_{M_h-2}}}, t_f|h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}})V_a(\chi^{\circ}_{1_{h_{M_h-1}}}, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{N_{h_{M_h-1}}}) \\
&V_a(\chi^{\circ}_{N_{h_{M_h-1}}}, s_{N_{h_{M_h-1}}}, t_f)) \geq p(0_{h_{M_h-1}})V_h(t_{x_{h_{M_h-2}}}, t_f|h_{M_h-1}) + p(1_{h_{M_h-1}}) \\
&(\delta_{1_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{1_{h_{M_h-1}}}|h_{M_h-1}) + (1-\delta_{1_{h_{M_h-1}}})V_f(\chi^{\circ}_{x_{h_{M_h-2}}}, s_{1_{h_{M_h-1}}}, t_f)) \\
&+ ... + p(N_{h_{M_h-1}})(\delta_{N_{h_{M_h-1}}}V_h(t_{x_{h_{M_h-2}}}, t_{N_{h_{M_h-1}}}|h_{M_h-1}) \\
&+ (1-\delta_{N_{h_{M_h-1}}})V_f(\chi^{\circ}_{x_{h_{M_h-2}}}, s_{N_{h_{M_h-1}}}, t_f)) \\
&= V_f(\chi^{\circ}_{x_{h_{M_h-2}}}, s_{x_{h_{M_h-2}}}, t_f) \geq V_f(\chi^{\circ}_{x'_{h_{M_h-3}}}, s_{x_{h_{M_h-2}}}, t_f)
\end{aligned}
\tag{4.11}
$$

Applying the same process iteratively, it can be seen that

$$
\begin{aligned}
&V_a(\chi^{\circ}_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) \geq V_f(\chi^{\circ}_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) \\
&\geq V_f(\chi^{\circ}_{x'_{h_{k-2}}}, s_{x_{h_{k-1}}}, t_f) \\
&... \\
&V_a(\chi_f, s_{t_i}, t_f) \geq V_f(\chi_f, s_{t_i}, t_f)
\end{aligned}
$$

Thus it proves that given an optimal policy of FSOA $\hat{\pi}^*_f(s_t, \chi_f, h_t)$, there will always be a heuristic reactive policy $\hat{\pi}_a(s_t, \chi_f, h_t)$ to achieve at least better estimated objective value, which proves the theorem. □

Theorem 4.7 shows that the heuristic reactive policy $\hat{\pi}_a$ is also sum-optimal, by comparing with the policy of FSOA. The hybrid policy and the heuristic reactive policy both try to improve the performance compared with the policy of FSOA, by introducing branchings in different ways. Their performances will be compared through simulation.

4.3.4 Monte-Carlo Estimation of Objective Value

For all the above strategies: the policy of FSOA $\hat{\pi}_f$, the hybrid policy $\hat{\pi}_m$, and the heuristic reactive policy $\hat{\pi}_a$, given a base trajectory, the branching function can uniquely and deterministically define all possible branchings. Thus for all there policies, they are clearly defined by their base trajectory. Then, there is a mapping from a base trajectory, initial condition, and time horizon, to the objective value of the respective policy:

$$V(\hat{\pi}(s_t, \chi, h_t), s_{t_i}, t_f) = \begin{cases} V_f(\chi, s_{t_i}, t_f) \text{ Policy of Fixed Sequence of Actions;} \\ V_m(\chi, s_{t_i}, t_f) \text{ Hybrid Policy;} \\ V_a(\chi, s_{t_i}, t_f) \text{ Heuristic Reactive Policy;} \end{cases} \tag{4.12}$$

We can see that, one unresolved problem of the policy planning is to calculate the objective value of each policy, given the base trajectory. For the policy of FSOA, as mentioned in Section 4.3.3.2, the objective value $V_f(\chi, s_{t_i}, t_f)$ can be calculated directly. For the heuristic reactive policy, the action taken at each time step is dependent on the current state. To predict the action at certain state and time step, the branching function needs to be applied, which is a forward induction process. The evolution of the probability distribution of the belief state $p(s_t|\hat{\pi}_f, s_{t_i})$, and the objective value $V(\hat{\pi}_f, s_{t_i}, t_f)$ can be then calculated. For the hybrid policy, it follows the same process but differs in that, the branching function is a backward induction, which should be much more complicated.

Although the branching has been simplified in both hybrid policy and heuristic reactive policy, it still causes huge amount of computation, because the number of calculation for branching function grows geometrically with the depth of branching. Hence for the planning of there two policies, two Monte-Carlo sampling methods are proposed to estimate the objective value.

Monte-Carlo Sampling for Hybrid Policy

For the hybrid policy, certain number of samples in each level of branching are taken. Assume that the maximum number of branchings for the hybrid policy is K. Let $m(k)$ be the function specifying the number of samples at each branch at each level, where $k \in [0, K]$. Using the same notation in the proof of Theorem 4.5, let $V_m(\chi, s_{t_i}, t_f) = V(\hat{\pi}_m, s_{t_i}, t_f)$ be the objective value of applying hybrid policy $\hat{\pi}_m(\chi, s_{t_i}, t_f)$, given initial state s_{t_i} and end of time horizon t_f. Then $V_m(\chi, s_{t_i}, t_f)$ can be constructed as follows, including all the possible branchings

$$
\begin{aligned}
&V_m(\chi, s_{t_i}, t_f) = p(0_\phi)V_h(t_i, t_f|\phi) + p(1_\phi)(\delta_{1_\phi}V_h(t_i, t_{1_\phi}|\phi) \\
&+ (1-\delta_{1_\phi})V_m(\chi^\circ_{1_\phi}, s_{1_\phi}, t_f)) + ... + p(N_\phi)(\delta_{N_\phi}V_h(t_i, t_{N_\phi}|\phi) \\
&+ (1-\delta_{N_\phi})V_m(\chi^\circ_{N_\phi}, s_{N_\phi}, t_f)); \\
&... \\
&V_m(\chi^\circ_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) = p(0_{h_k})V_h(t_{x_{h_{k-1}}}, t_f|h_k) + p(1_{h_k})(\delta_{1_{h_k}} \\
&V_h(t_{x_{h_{k-1}}}, t_{1_{h_k}}|h_k) + (1-\delta_{1_{h_k}})V_m(\chi^\circ_{1_{h_k}}, s_{1_{h_k}}, t_f)) + ... + p(N_{h_k}) \\
&(\delta_{N_{h_k}}V_h(t_{x_{h_{k-1}}}, t_{N_{h_k}}|h_k) + (1-\delta_{N_{h_k}})V_m(\chi^\circ_{N_{h_k}}, s_{N_{h_k}}, t_f)); \\
&...k \in [1, M_h - 1]
\end{aligned}
\tag{4.13}
$$

where $M_h \leq T/\Delta T \leq K$ is the maximum levels of branchings in current branch, given the history h. $\delta_{x_{h_k}} = (t_{x_{h_k}} - t_{x'_{h_{k-1}}})/(t_f - t_{x'_{h_{k-1}}})$, $\chi^\circ_{x_{h_k}} = f_m(s^\circ_{x_{h_k}}, \chi^\circ_{x'_{h_{k-1}}}, h_k)$.

At level $k \in [1, M_h - 1]$, for a branch with history h_k, $m(k)$ cases of samples of the possible branchings $Sp(h_k) = \{x_{h_k} : x \in [0, N_{h_k}]\}$ are taken along the base trajectory $\chi^\circ_{x_{h_{k-1}}}$, which is chosen with a probability of this branching $p(x_{h_k})$. Let $V_i(\chi^\circ_{x_{h_k}}, s_{x_{h_k}}, t_f)$ substitute the objective value of the sampled branch triggered by x_{h_k}.

Thus Equation 4.13 can be written as:

$$
\begin{aligned}
&V_i(\chi, s_{t_i}, t_f) = \sum_{x \in Sp(\phi)} (\delta_{x_\phi}V_h(t_i, t_{x_\phi}|\phi) + (1-\delta_{x_\phi})V_i(\chi^\circ_{x_\phi}, s_{x_\phi}, t_f))/m(0); \\
&... \\
&V_i(\chi^\circ_{x_{h_{k-1}}}, s_{x_{h_{k-1}}}, t_f) = \sum_{x \in Sp(h_k)} (\delta_{x_{h_k}}V_h(t_{x_{h_{k-1}}}, t_{x_{h_k}}|h_k) + \\
&(1-\delta_{x_{h_k}})V_i(\chi^\circ_{x_{h_k}}, s_{x_{h_k}}, t_f))/m(k); \\
&...k \in [1, M_h - 1]
\end{aligned}
\tag{4.14}
$$

Thus the objective value of each branching, $V_m(\chi^\circ_{x_{h_k}}, s_{x_{h_k}}, t_f)$, is approximated by the sampling value $V_i(\chi^\circ_{x_{h_k}}, s_{x_{h_k}}, t_f)$. When $k = M_h - 1$, $V_i(\chi^\circ_{x_{h_{M_h-1}}}, s_{x_{h_{M_h-1}}}, t_f) = \max_\chi V_f(\chi, s_{x_{h_{M_h-1}}}, t_f)$. Then $V_i(\chi, s_{t_i}, t_f)$ can be calculated with backward induction. The value of the hybrid policy is then approximated as:

$$
V_m(\chi, s_{t_i}, t_f) = V_i(\chi, s_{t_i}, t_f) \tag{4.15}
$$

The sampling reduces the quantity of branchings, thus making the computation in planning feasible. Also, by letting $m(k)$ to have different numbers of samples in each level, more importance can be given to future steps which are more immediate and with fewer branches to consider, than the further steps with more branches. Thus it can concentrate the samples and the computational power on more important part of the decision tree.

Monte-Carlo Sampling for Heuristic Reactive Policy

For the heuristic reactive policy, the calculation of branching is a forward induction method, which is much faster, thus the sampling can be simpler. Sampling is not done hierarchically in each level. Instead, m cases of simulations are conducted. In each simulation, starting from the given initial state s_{t_i}, the agent implements the policy $\hat{\pi}_a(s_t, \chi, h_t)$ until the end of time horizon t_f. Let each event happen stochastically based on its probability, and the agent reacts according to the policy. In each sample $i = 1, ..., m$, the achieved hindsight objective value $V_i(\hat{\pi}_a, s_{t_i}, t_f)$ can be computed based on the events occurred and the actions taken. Then the objective value $V(\hat{\pi}_a, s_{t_i}, t_f)$ can be approximated by:

$$V(\hat{\pi}_a, s_{t_i}, t_f) = \sum_{i \in [1,m]} V_i/m \tag{4.16}$$

From the above steps, it has built the sampling methods of estimating the objective value of the hybrid policy and the heuristic reactive policy. Then, the mapping described in Equation (4.12) can be obtained. Given such mapping, the policy planning equation

$$\hat{\pi}^* = argmax_{\hat{\pi}} V(\hat{\pi}, s_{t_i}, t_f) \tag{4.17}$$

has become

$$\chi^* = argmax_{\chi} V_{f/m/a}(\chi, s_{t_i}, t_f) \tag{4.18}$$

Then the policy planning problem has been simplified to be a path planning problem. It needs to search the best base trajectory χ^*, with a highest objective value of corresponding strategy.

4.4 Path Planning based on Simulated Annealing

For the path planning problem defined in Equation (4.18), a conventional method is to do a tree search [100], in which the trajectory is treated as a sequence of actions, rooted at the initial location. Then the possible solution paths are inducted as a tree, branching at each step according to different actions to take. Such tree search can be rigorous and can precisely find the best path. However, this method is unintuitive, making it difficult to be simplified with domain knowledge. The branching make it unscalable w.r.t the time horizon.

Therefore, the sampling method is taken as solution. A candidate path is proposed initially, and random modifications on such path are imposed iteratively, until the path being satisfactory. Doing improvement on an existing path can be more intuitive, thus is easier to incorporate heuristics. By limiting the number of improvements to make, the computational time can be constrained.

4.4.1 Further Simplification

Before designing the path planning algorithm, a further assumption is made to facilitate the planning.

5. **Trajectory Planning Constraint**. Assume that the vertices of a planned path can only be $C_s \bigcup \{\hat{\mathrm{x}}_\lambda(t) : \lambda \in \Lambda_t\}$, which are enough to cover the whole environment without undermining performance. The former set of cells are called search cells, and the later are called monitoring cells.

This simplification is to limit the locations on the map to consider, thus making the path planning more efficient.

4.4.2 Candidate Trajectory Mutation

The concept of path planning based on sampling is a path improvement process, which does mutations on a candidate path, trying to improve the reward. Let $\hat{\chi} = M(\chi)$ be the mutation function for a trajectory. Four kinds of mutations are designed, as inspired by [101]:

1. **Add**: at one position of χ, add a new node;
2. **Prune**: prune one node from χ;

3. **Swap**: swap the position of two nodes in χ or swap one node in χ with a new location;

4. **Null**: keep χ unchanged.

mutation 1-3 are shown in Figure 4.5.

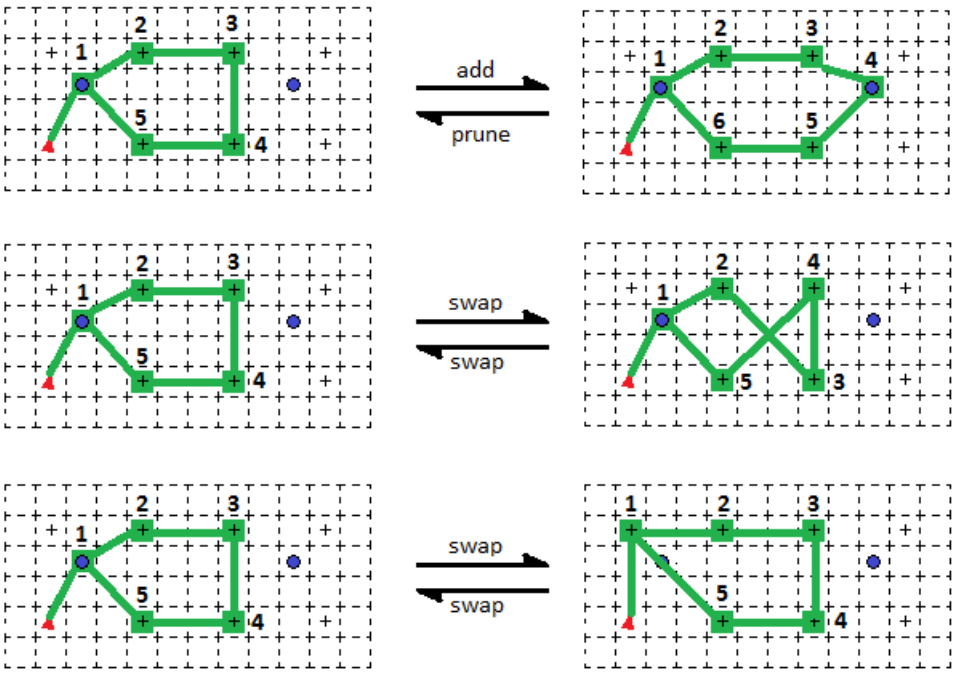


FIGURE 4.5: mutations on trajectory

The red triangle is the current agent location. Green vertices and lines denote the planned trajectory. The numbers show the sequence of nodes. The cells with a plus signs are search cells. The cells with blue solid circles are monitoring cells.

4.4.3 Simulated Annealing Algorithm for Path Planning

With the mutation function, the path χ can be planned by a Path Planning algorithm based on Simulated Annealing (Algorithm 5) [102, 103]. Simulated Annealing is widely used in path planning and can effectively avoid local minima [103–106].

Then, the reactive policy can be planned by above steps, which can be executed by the agent for SSM mission.

Algorithm 5: Path Planning Algorithm based on Simulated Annealing

initialization;
$\chi = \{x'_\rho(t) : t = t_i, t_i + 1, ..., t_f\}$,$T_e = T_{e0}$,$k_B = const$,$V_c = 0$
while $T_e \geq T_{default}$ **do**
  $\hat{\chi} = M(\chi)$. $V_\rho = -V(\pi_a, s_{t_i}, t_f)$, $E = |V_\rho - V_c|$
  if $V_\rho > V_c$ **then**
    $p = exp(-E/k_B T_e)$
    if $random(0, 1) \leq p$ **then**
      accept = true
    else
      accept = false
    end
  else
    accept = true
  end
  if *accept* = *true* **then**
    $V_c = V_\rho$, $\chi = \hat{\chi}$
  end
  Lower the temperature T_e
end
Output χ

4.5 Simulation Evaluation and Validation

4.5.1 Case Study

Consider a $100m \times 100m$ square environment ς, which is discretized into 25×25 cells. The agent sensor can cover 5×5 cells. There are 5 unknown targets and 1 pursuer scattered in the environment. For each time step $\Delta T = 0.2s$, there will be $p_s = 80\%$ probability that a target will stay within the current location. The pursuer can move at speed $V_\rho = 20m/s$. The agent will plan and execute a policy for the SSM task with a time horizon $T = 10s$, which can be the policy of FSOA, the hybrid policy with $K = 1$ or the heuristic reactive policy. The number of steps in the planning Horizon is $T/\Delta T = 50$. When a contingency state s° is reached, or when it has been after T_p long time since last planning, a replanning will be triggered. Set $T_p = 5s < T$ to make the planning more adaptive to environmental changes. The initial target probability distribution $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$ is uniform within the environment, and targets are randomly scattered initially. The actions of the three policies are studied in this Section.

Figure 4.6 to 4.14 are the snapshots of simulation with the three policies.

The polylines with arrows are the plans of base trajectory. It can be seen from Figure 4.6 to Figure 4.8 that, when there is an area with high distribution of unknown targets, the agent will sweep that area to search. Figure 4.9 to Figure 4.11 show that when

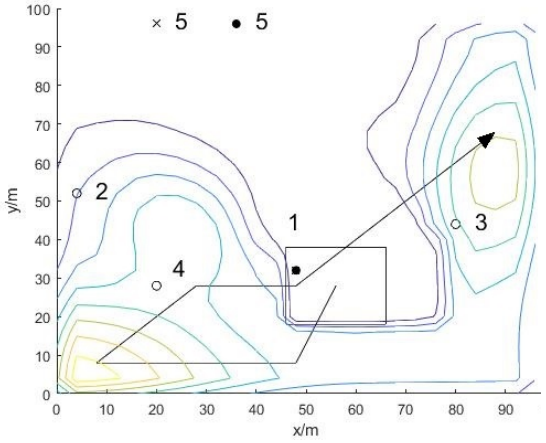


FIGURE 4.6: search (FSOA)

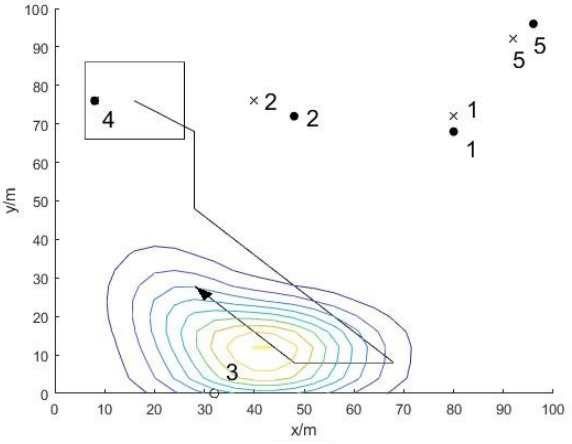


FIGURE 4.7: search (hybrid)

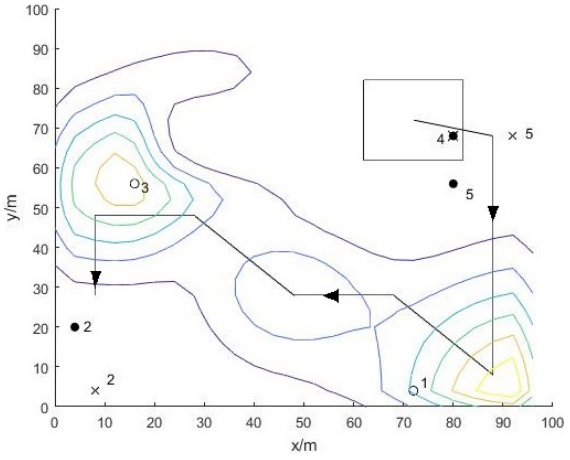


FIGURE 4.8: search (heuristic reactive)

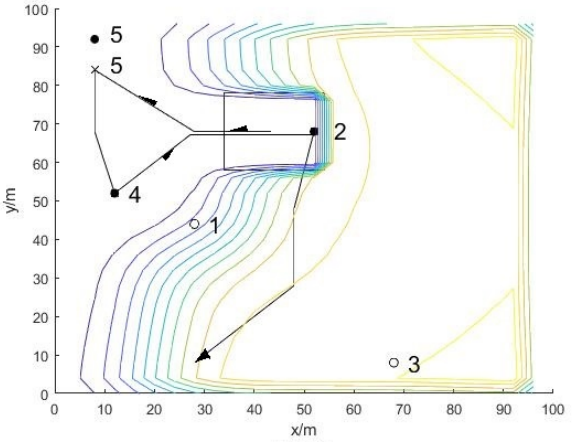


FIGURE 4.9: SSM (FSOA)

some targets are known to be nearby and there is likely to be an unknown target in the neighbouring area, the agent may try to explore the neighbouring unknown area and traverse the known targets, thus combining search and monitoring in the same path. Figure 4.12 to Figure 4.14 shows that when the monitoring is saturated, which is when there are some known targets nearby but there is unlikely to be unknown targets in vicinity, the agent will focus on traversing nearby known targets back and forth.

Figure 4.15 to Figure 4.17 illustrate the *belief probability* of each target and the overall reward of the SSM mission at each time step of a case study, which are $\tilde{B}_\lambda$ and $R = \sum_{\lambda \in \Lambda} \tilde{B}_\lambda$. The *belief probability* of a target increases to 1 when it is detected, and drops to 0 when it is lost. The *belief probability* degrades gradually when the target is not being measured. It can be seen that, with all three policies, every target can be detected during the simulation. Most of them can be maintained a high *belief probability* for several non-continuous periods, and can be re-detected intermittently after being unattended. The negative spikes show that the targets may get lost when the agent tries to re-detect

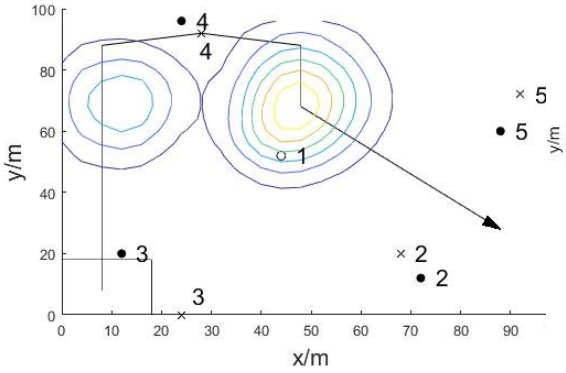


FIGURE 4.10: SSM (hybrid)

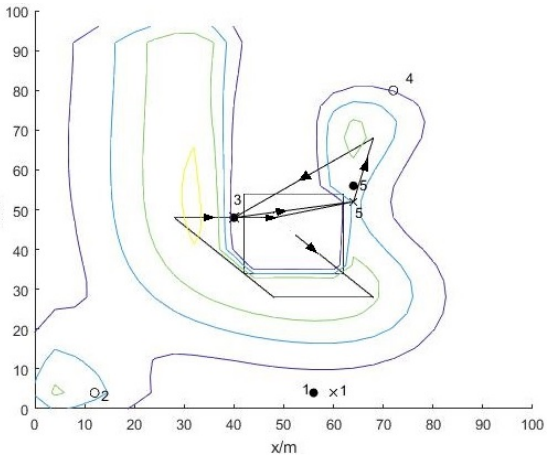


FIGURE 4.11: SSM (heuristic reactive)

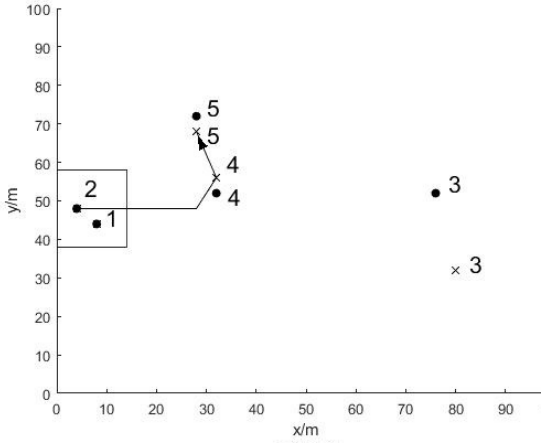


FIGURE 4.12: monitoring (FSOA)

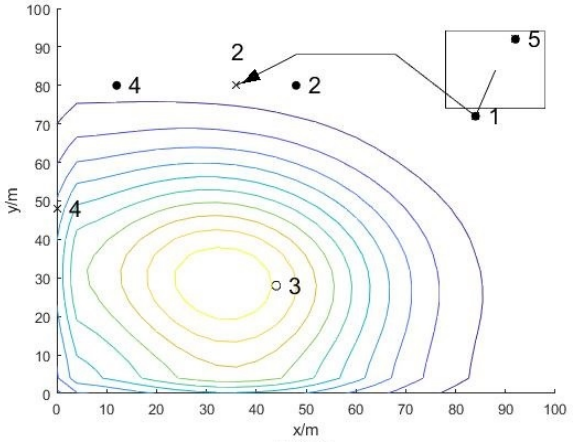


FIGURE 4.13: monitoring (hybrid)

them, but they will soon be retrieved. The overall reward is increased shortly after the simulation starts, and is kept above a certain level with small fluctuations.

The case study qualitatively shows that, all three polices appear to show the similar pattern of behaviour. By dynamically combine search and monitoring, the agent can efficiently search for hidden targets, and preserve the *belief probability* of as many targets as possible, thus to maintain a high objective value.

4.5.2 Comparative Study

The patterns of behaviours appear to be analogous for different policies, so this section does the quantitative study of the performance of proposed policies. The strategy are compared by the average reward that they can achieve, and the average computation time of each planning.

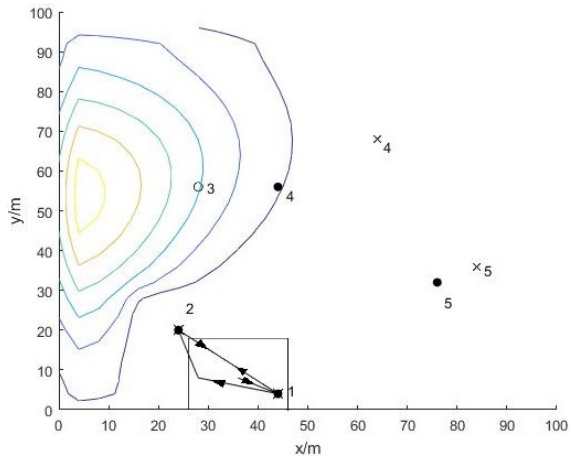


FIGURE 4.14: monitoring (heuristic reactive)

With the same set-up of environment in the case study, for each strategy, the scenarios are studied, with n = 2, 3, 5 and 7 targets and with $p_s = 60, 70, 80\%$. For each scenario, 100 cases of simulations are done for 200 seconds long each. In each case, the average reward at every time step is taken as the reward achieved in that case. The reward of a scenario is the average reward of every case. The computation time of each planning is recorded and averaged for each scenario as well.

The cases with imperfect sensor are also considered, where at each time step, for the sensing of each target, there would be 0.2% chance of false positive or 5% chance of false negative. Figure 4.18 shows the performances in each scenarios by each policy. Each simulation is done by one core of E5 2650V2 processor (2.6 GHz).

It can be seen from Figure 4.18 that, in most scenarios, the rewards of both the heuristic reactive policy and the hybrid policy are significantly better than that of the FSOA. It proves that, if the future contingencies and corresponding reactions are considered during planning, the agent can make better decision about future actions, which is consistent with Theorem 4.7 and 4.5.

But also, Figure 4.18 shows that the reward of the heuristic reactive policy is better than the hybrid policy in almost every scenario. It proves that the approach taken by the heuristic reactive policy, which considers all the possible branchings by having a heuristic branching function to do forward induction, can achieve better performance than the more rigorous method of hybrid policy, when it does not have enough levels of branching.

The average computation time in different scenarios with different planning methods is in Table 4.1.

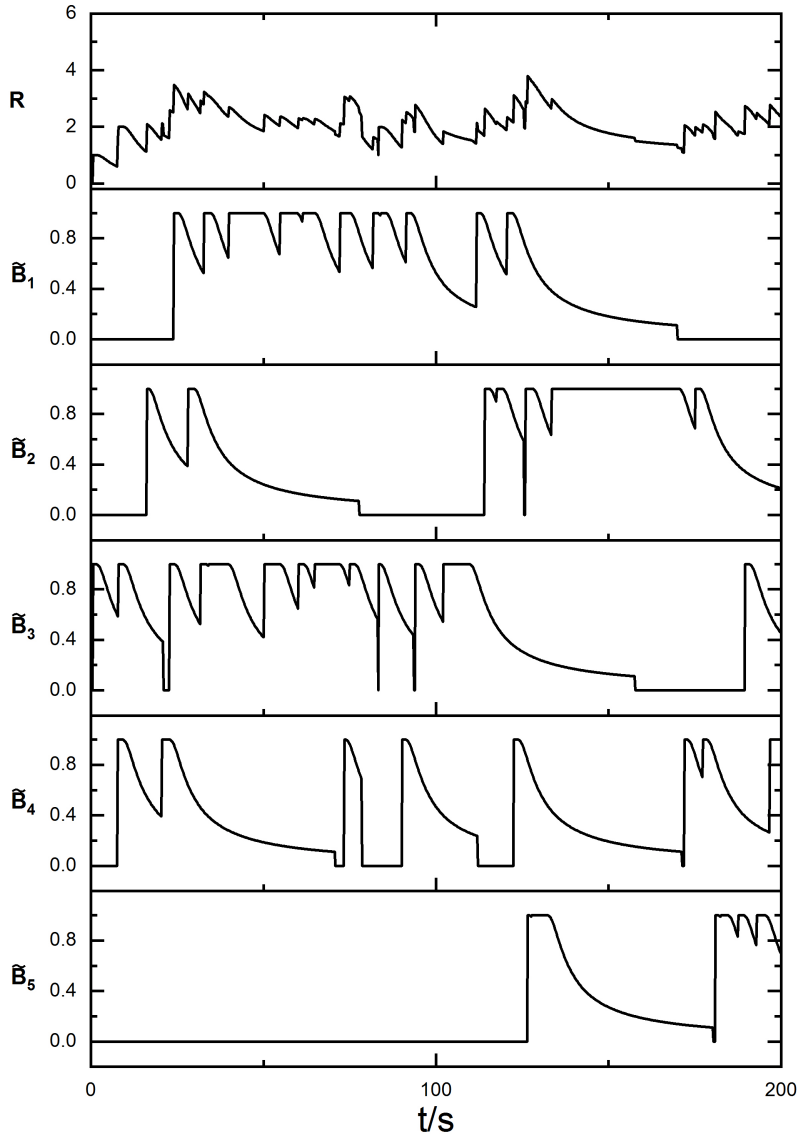


FIGURE 4.15: *belief probability maintenance (FSOA)*

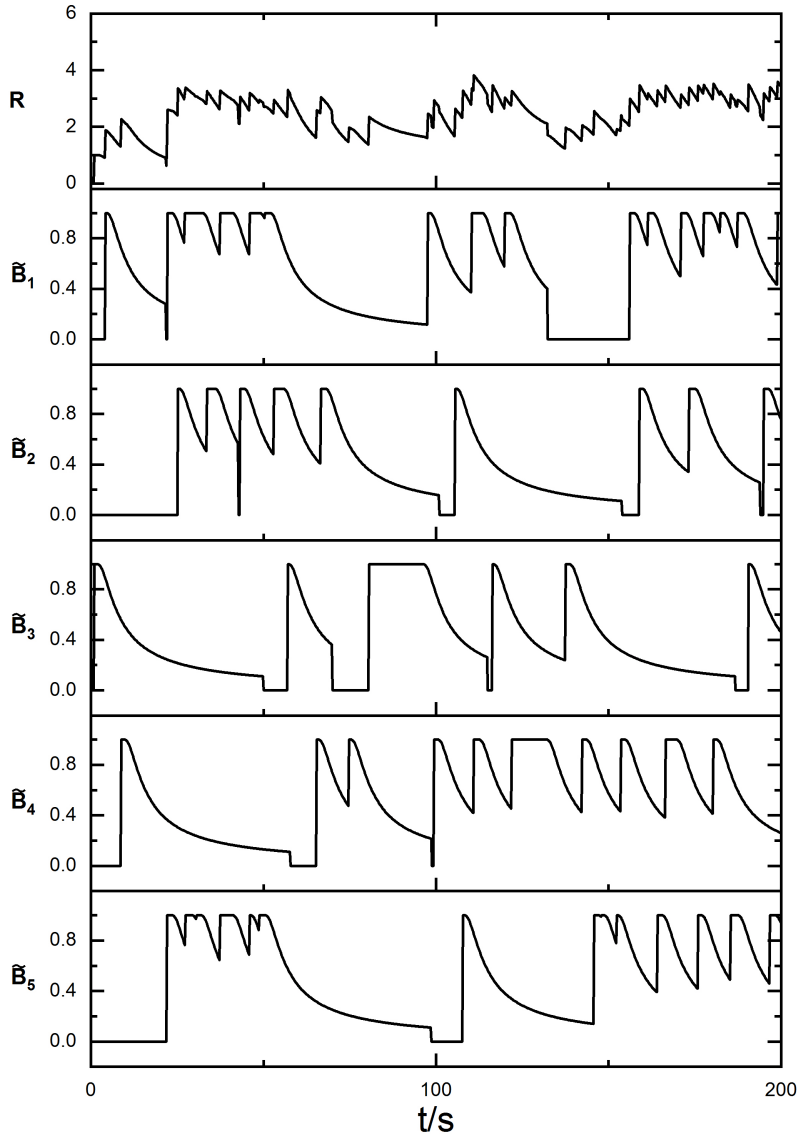


FIGURE 4.16: *belief probability maintenance (hybrid)*

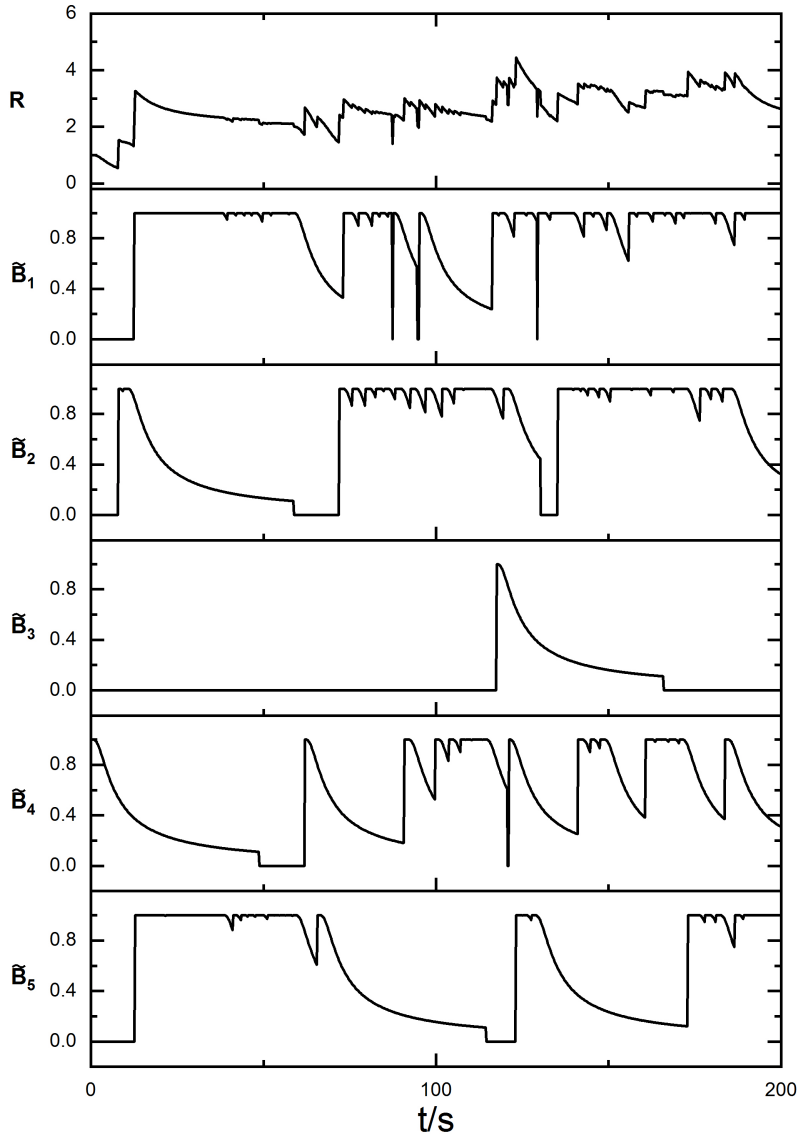


FIGURE 4.17: *belief probability maintenance (heuristic reactive)*

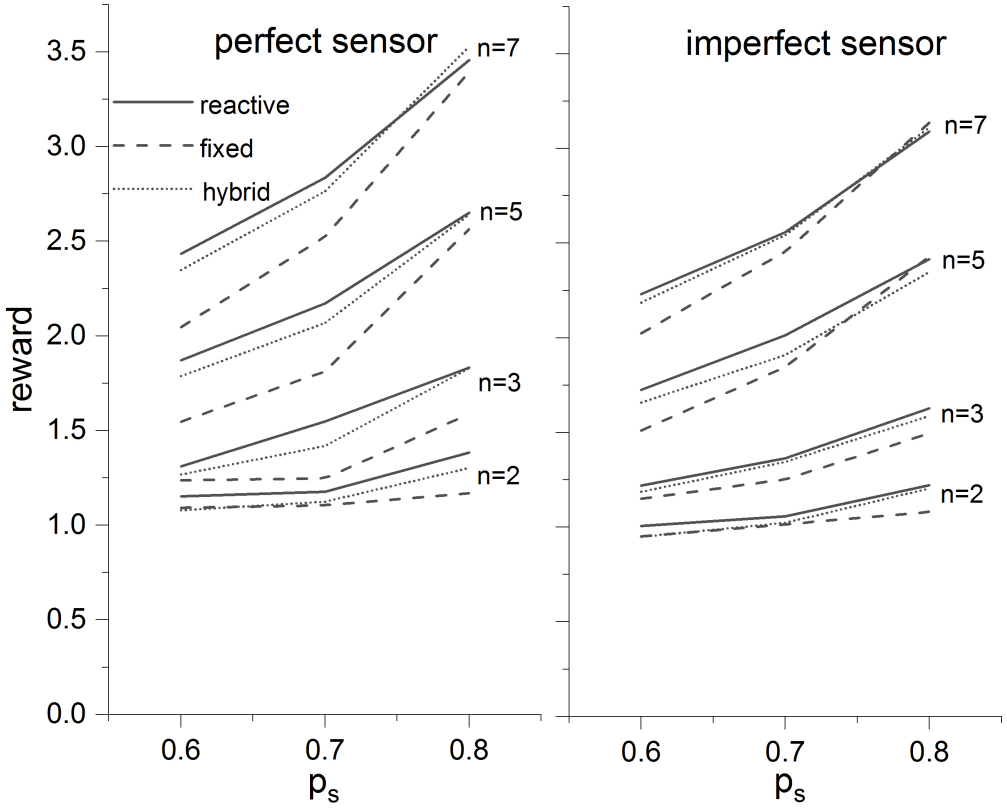


FIGURE 4.18: reactive policy vs. hybrid policy vs. fixed sequence of actions

number of targets	2	3	5	7
FSOA	0.01s	0.01s	0.01s	0.01s
hybrid Policy	2.72s	3.65s	5.71s	6.95s
heuristic reactive Policy	0.14s	0.19s	0.22s	0.25s

TABLE 4.1: computation time for different policy planning

We can see from Table 4.1 that, for all scenarios, the planning of heuristic reactive policy takes 0.2 seconds in average. The hybrid policy takes 4.76 seconds, even when the branching level K is only 1. Considering that the planning horizon $T = 10s$, this further shows the advantage of the heuristic reactive policy over the hybrid policy, in terms of the practicability. Both policies are both much slower than the policy of FSOA, which takes 0.01 seconds in average. Nevertheless, the speed of heuristic reactive policy is still practical for real time implementation.

It is also shown from Figure 4.18 that, in the case of imperfect sensor, there will be a decrease in the performance of all approaches. However, this can be improved by introducing sensor filtering to reduce the influence of false measurement.

To explain the advantage of the heuristic reactive policy and hybrid policy over the policy of FSOA, the following case is studied. It is a snapshot of a situation in the simulation, where there are only two known targets and $p_s = 80\%$. For the same situation, policies are planned using all three proposed planning methods. The base trajectories planned by these methods are shown in Figure 4.19

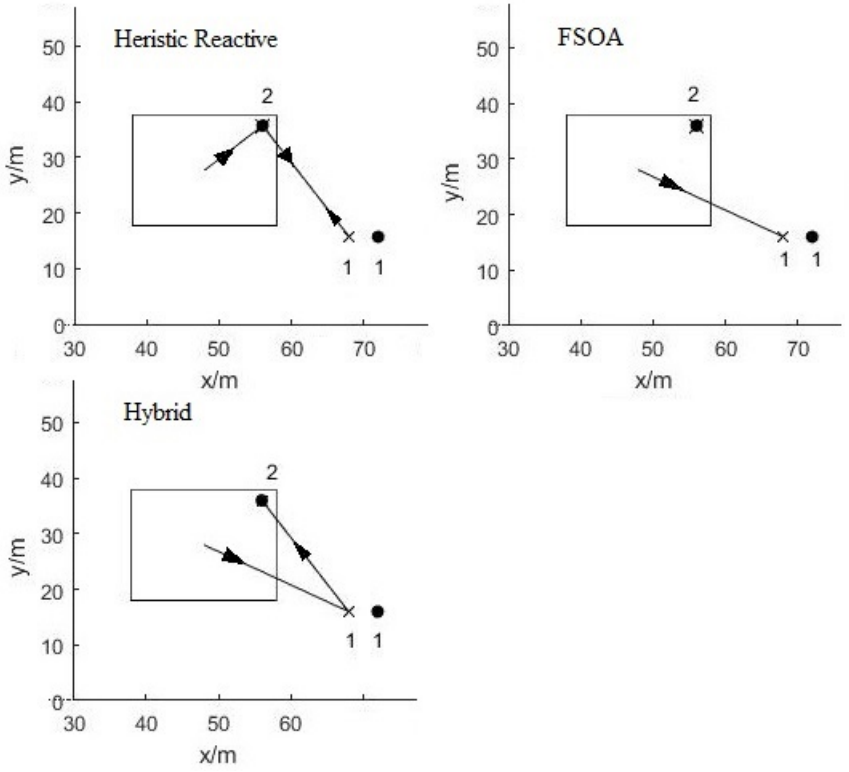


FIGURE 4.19: base trajectory planned by the heuristic reactive policy (left up), the policy of FSOA (right up), and the hybrid policy (left down).

In this case, the policy of FSOA keeps the robot to follow only one target, with an estimated objective value of 1.70. The hybrid policy lets the agent visit target 1 and 2, then keep following target 2, with an estimated objective value of 1.76. However, the heuristic reactive policy planning drives the robot to go back and forth between two known targets, with a better objective value of 1.92. The policy of FSOA does not choose the back-and-forth route, because if it follows such a fixed route, the agent will not react if one target is lost, and will still go back and forth. Then the remaining target will always have a chance to escape between each visit. However, with the heuristic reactive policy, if a target is lost, the agent will go back and focus on monitoring the remaining one, which is more rational and is with higher estimated objective value. For the hybrid policy with a branching factor $K = 1$, it allows only one level of branching, which is not capable of reasoning about the future that far ahead, thus it does not allow the agent to go back to target 1 after visiting target 2. We can see that, while the policy of FSOA

tends to be conservative when there is a risk, the heuristic reactive policy allow the agent to make more sensible decisions. The hybrid policy is an intermediate between them.

4.5.3 Considering the Manoeuvrability of the Pursuer

All the previous research is based on the assumption that the turning radius of the agent is ignored. To validate the results with a more realistic agent model, the Dubin Vehicle model is considered for the agents, where there is a maximum lateral acceleration. Assume a constant speed of the agent, the limit on the lateral acceleration can be equivalent to a minimum turning radius. Assume that during the strategy planning, the agent still assumes its perfect agility. But in the simulation, the motion of the agent is limited by a minimum turning radius $r_c = 5m$. This realistic limitation is illustrated by Figure 4.20

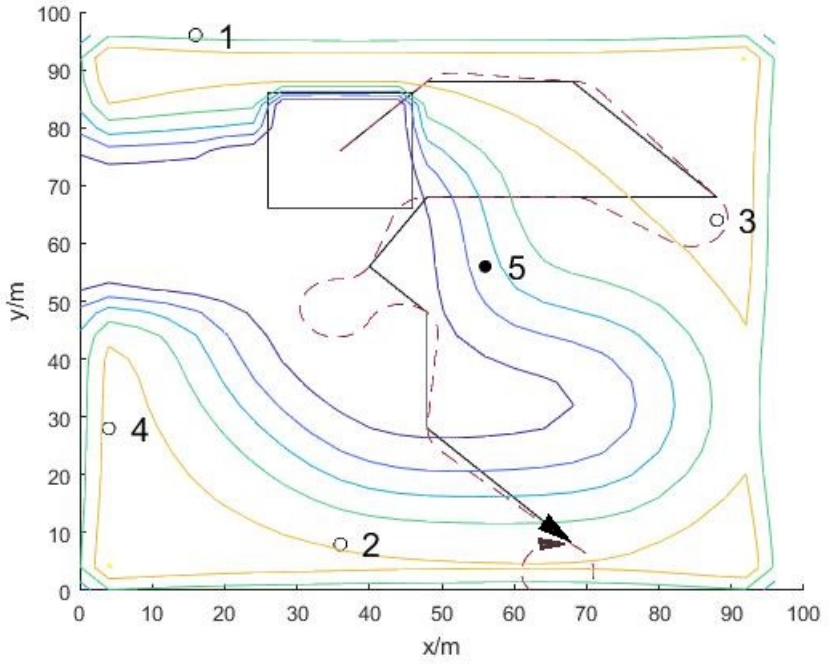


FIGURE 4.20: the planned base trajectory of the agent (solid line) and the actual achievable trajectory (dash line)

With such modification in the simulation, the comparative study for all scenarios in Section 4.5.2 is done again, and the result is shown in Figure 4.21.

We can see that, in a more realistic situation, the rewards of both the heuristic reactive policy and the hybrid policy are still dramatically better than that of the FSOA. In this realistic simulation, the advantage of the performance of the heuristic reactive policy is not as obvious compared with the hybrid policy. However, for the heuristic reactive

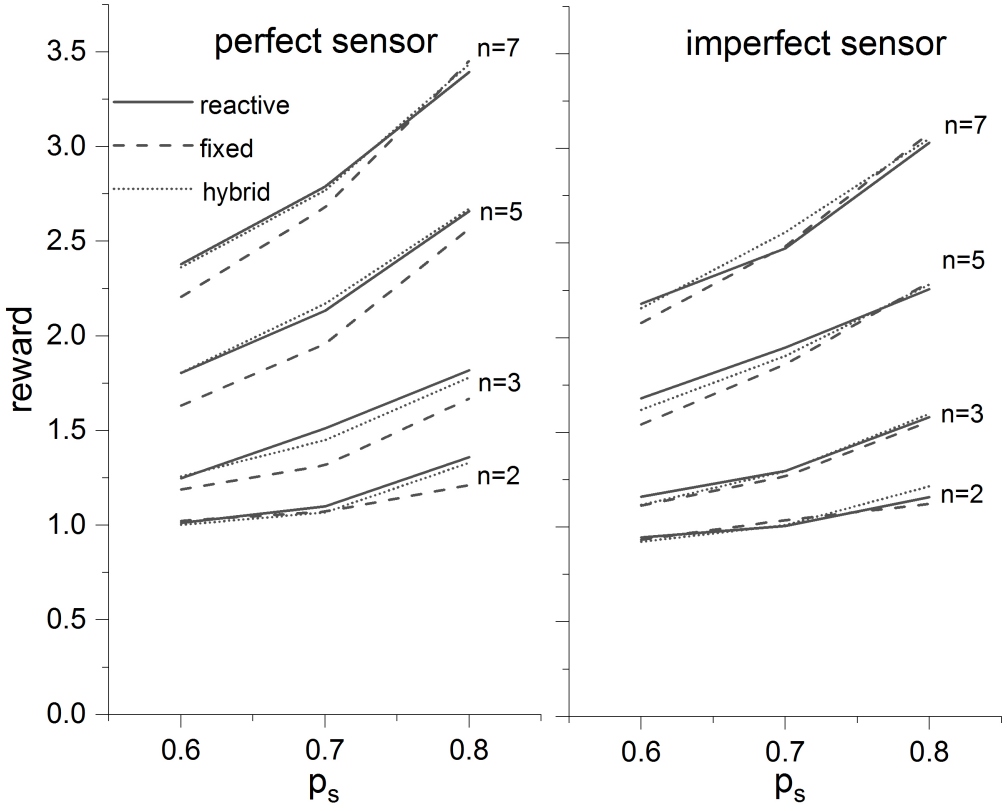


FIGURE 4.21: reactive policy vs. hybrid policy vs. fixed sequence of actions, with $r_c = 5m$

policy, the much faster computation speed makes it a more practical choice for strategy planning, given that the planning horizon is 10 seconds. Therefore, the heuristic reactive policy planning is chosen to be the solution of the single-pursuer SSM of the randomly moving targets.

4.5.4 Exploring the limitations of SSM

The previous sections have validated the efficiency of SSM in moderate scenarios. In this section, the simulations are expanded to some more extreme situations, to find out the practical limitations on SSM. We expand the scope of simulation in two dimensions separately: the activity level of targets and the size of environment. Compared with the simulation in Section 4.5.2, we either expand the p_s to $20, 30, 40, 50\%$, or expand the arena to $140m \times 140m$ and $180m \times 180m$. The results are shown in Figure 4.22 and 4.23:

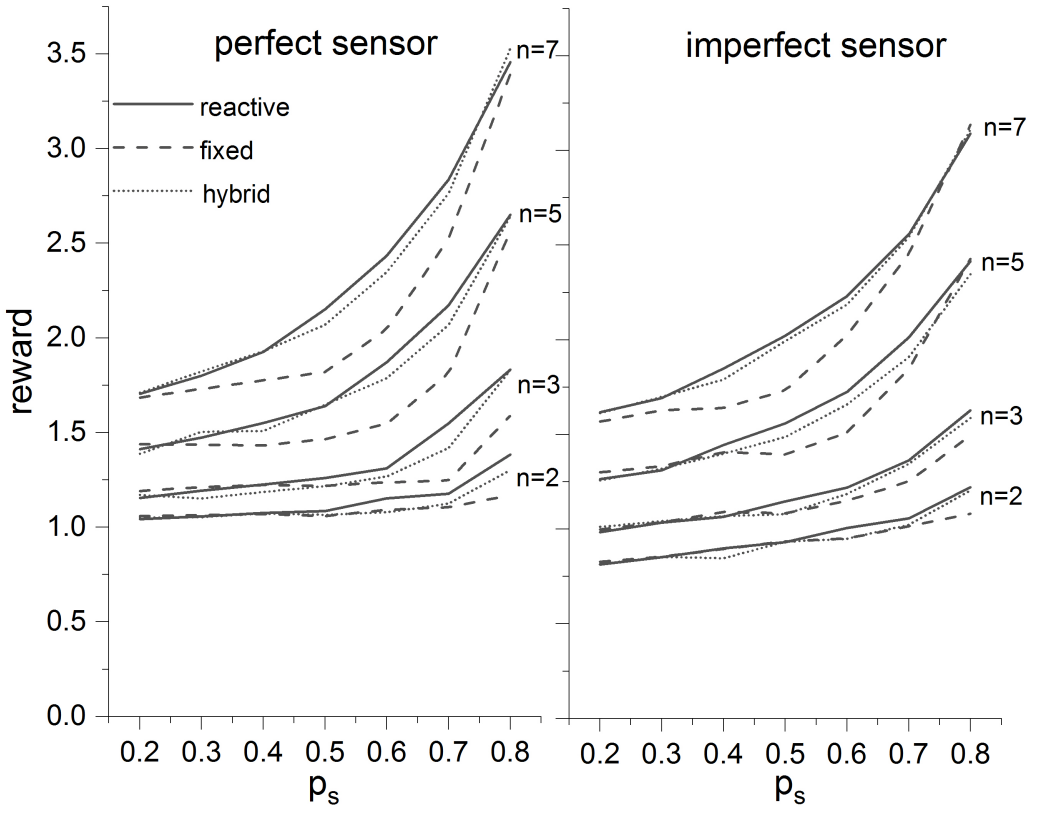


FIGURE 4.22: reactive policy vs. hybrid policy vs. fixed sequence of actions, with environment size equals $100m \times 100m$, and with expanded range of p_s

From Figure 4.22 to 4.23, we can see that, with targets being more active or with the environment being bigger, it is more difficult for the agent to do SSM. In Figure 4.22, when $p_s \leq 40\%$, the heuristic reactive policy has only marginal advantage against FSOA and hybrid policy. Besides, in these scenarios, the reward of SSM decreases to be only slightly higher than 1, which is the reward of monitoring one single target. In 4.23, when the environment is $180m \times 180m$, we can also see that the heuristic reactive policy only has trivial advantage, and the performance of SSM degrades to be only slightly better than having one target under monitoring.

Thus we can see that, with current capabilities of the agent (size of sensor footprint and maximum velocity), when $p_s \leq 40\%$ or when the size of environment is $180m \times 180m$, the performance of SSM reaches its limit. In these cases, it is more favourable for the agent to keep monitoring the first target it finds, in which each policy it takes does not make a difference.

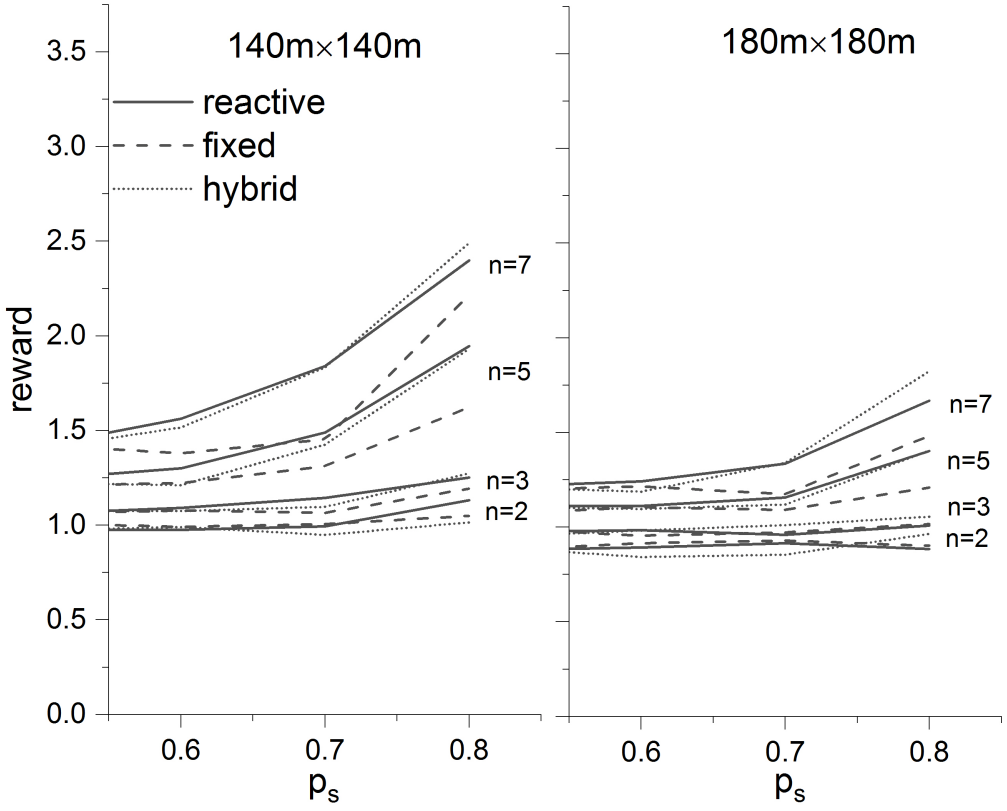


FIGURE 4.23: reactive policy vs. hybrid policy vs. fixed sequence of actions, with environment size equals $140m \times 140m$ and $180m \times 180m$

4.6 Conclusion

For the SSM between a single pursuer and multiple targets, the problem is formulated as a POMDP. By building an appropriate objective function, the search and monitoring are combined under a united reward. The online solution is chosen, in order to let the planning be scalable to the size of the problem, and be adaptive to environmental changes. To tackle the computational intractability, a novel policy reconstruction method is proposed, to allow building a heuristic structure of policy. Then three sub-optimal policies are designed, considering the trade-off between the performance and the computational efficiency.

The case study simulation result shows that, all three proposed policies can effectively search for hidden targets in an initially unknown environment, and can maintain the surveillance of them, with a moderate computational cost. Whenever the monitoring capability is not saturated, the agent will try to find more targets without losing current

known ones. In the comparative study, it is validated that both the hybrid policy and the heuristic reactive policy works better than the conventional policy of FSOA. It also shows that the heuristic reactive policy is better than the hybrid policy in terms of the performance and the computational efficiency. In the more realistic situation when the turning radius of the pursuer is considered, the above advantage of the heuristic reactive policy still uphold. Thus the heuristic reactive policy is chosen to be the solution of the single pursuer SSM of the randomly moving targets, for its better performance and practical computation speed. In the later multi-pursuer SSM in Chapter 6, the concept of heuristic reactive policy will be taken as the foundation.

The limitations of SSM of randomly moving targets are studied in Section 4.5.4. The activity level of targets and the size of environment are expanded to bigger ranges, which indicate the scenarios when SSM is not practical any more. However, as shown in Section 4.5.1 and 4.5.2, under moderate conditions, SSM is still efficient and the advantages of heuristic reactive policy still holds.

Chapter 5

Simultaneous Search and Monitoring of Evasive Targets

The single pursuer SSM of randomly moving targets has been studied in Chapter 4. This chapter studies the single pursuer SSM of evasive targets. A rigorous problem formulation is done first, by building a Partially Observable Game Playing. Some precise solutions are introduced, and some other heuristic approaches are discussed as well. The precise solutions are intractable, but the conventional heuristic approaches are not suitable for intelligent evaders. To address this difficulty, an assumption about the information available to the targets is introduced and justified. Based on this assumption, the game playing can be simplified to a dynamic guaranteed search, which is much easier to solve. A policy planning approach is then proposed for the pursuer to achieve the SSM. The SSM is demonstrated in simulation. The performance of the dynamic guaranteed search is compared with a conventional guaranteed search method, and showed superior performance. The UAV model is also considered in the final section.

5.1 Formulating Partially Observable Game Playing

For the SSM of randomly moving targets studied in Chapter 4, the target behaviour is independent from the actions of the pursuers. Thus with a known stochastic motion model, the target location can be estimated by a probability distribution. The evolution of the game can be predicted w.r.t the actions of the pursuer, by a forward induction. However, when the targets can sense the location and action of the pursuer and evade detection proactively, it becomes a two-sided search and pursuit evasion game. The decision trees of both sides become a combined decision tree, making the planning of both sides coupled. Each side of the game needs to search through the whole combined

decision tree to predict the behaviour of the opponent, and evaluate its own strategy. This coupling make it much more difficult to solve two-sided search and pursuit evasion problem compared with the one-sided problem.

For the SSM in this work, considering the fact that the agents can have only partial observability of the environment, and that the pursuer and evaders have opposing goals, the Partially Observable Stochastic Game (POSG) is the most suitable framework to formulate the problem. As introduced in Section 2.2.2, the POSG can be described in a tuple $\langle I, S, \{A_i : i \in I\}, \{\Omega_i : i \in I\}, T, \{O_i : i \in I\}, \{R_i : i \in I\}\rangle$ [87], where

1. I is a finite set of players, which are the pursuer and the evaders. i is the label of a certain player, and $i \in I$;
2. S is a finite set of states of the world. The state $s_t = \{x_\rho(t), \{x_\lambda(t) : \lambda \in \Lambda\}\}$;
3. A_i is a finite set of actions of the player i;
4. $T : S \times \{A_i : i \in I\} \longrightarrow p(S)$ is the state-transition function, mapping from a previous world state and the joint actions of all the players, to a probability distribution of next world states;
5. $R_i : \boldsymbol{S} \times A_i \longrightarrow R_i$ is the reward function, mapping from a current world state and a player action to a immediate reward to that player;
6. Ω_i is a finite set of observation of the player i;
7. $O_i : \boldsymbol{S} \times A_i \longrightarrow p(\Omega_i)$ is the observation function, mapping from a current world state and an action of the player i, to a probability distribution of the observations of the player i.

For the evasive targets, because of the full observability that has been assumed, they can have access to the system state $s_t \in S$. For the pursuer, which can only measure within its sensor footprint, let $\hat{s}_t = \{\{\hat{\mathrm{M}}_\lambda(c, t|Y_t) : \lambda \in \Lambda\}, x_\rho(t)\}$ be its subjective state, which describes the understanding of the world from the perspective of the agent.

For the problem studied in this work, when both the pursuer and the targets can accurately control their motions, the state-transition function is deterministic rather stochastic. But because of the fact that the formulation of POSG includes the partial observability of the agent, it is still a suitable framework for studying this problem.

According to [32], Decentralized Partially Observable Markov Decision Process, which is a special type of POSG, is NEXP-complete. This indicates the difficulty of solving POSG. The solutions of POSG are review in Section 2.2.2. The solutions of POSG can

be offline or online, similar to POMDP. However, compared with POMDP, the solution of POSG needs double exponential time in the worst case [89]. For the same reason as discussed in Section 4.3.1, an offline solution is not favoured, because of its poor scalability and adaptability. To the author's knowledge, there has not been works on the online solution of POSG for non-cooperative game playing, and no work on robot pursuit evasion game is solved in the precise form of POSG.

In [30], the search evasion game is solved for a single time step, and the Nash Equilibrium is taken as the solution. Sharing the same problem with other myopic methods, such approach can not guarantee a long term global performance. In most of the robot pursuit evasion games, a common method is having a heuristic model to estimate the possible target behaviours.

5.2 Heuristic Models for Search and Pursuit Evasion Games

Search and pursuit evasion games normally have a long time horizon to consider, and a big area to play in. Because of the intractability of exact solution for POSG, very few work of search and pursuit evasion games apply such framework as solution. The most common method for solving search and pursuit evasion games is to propose a heuristic target model to approximate the evader strategy. By such means, the decision tree of the target can be replaced by an explicit policy. Then the agent does not need to calculate the possible rational plans of the target in order to evaluate the pursuit strategy, thereby disentangling the evader actions from the planning of pursuit policy.

Among these heuristic approaches, most works assume a pattern about how the targets will move away from the pursuer. In some works [34–36], a potential force is assumed to be imposed from each pursuer and obstacle. By applying the total potential force on each target, the targets move away from the pursuer, thus achieving the evasion. In [37, 38], a reactive rabbit is applied, in which the targets are driven away by the pursuer when they are within a certain distance.

For the evasive targets studied in this work, they are assumed to evade intelligently. Thus this work does not assume a specific heuristic pattern of the evader motions. Because when the evaders are intelligent, any pursuit strategy, which are based on a presumed target behaviour pattern, can be learned and taken advantage by the targets. Instead, the worst case assumptions are taken in this work, where all the possible reachable area of a target at each time step is calculated and a policy is planned which can guarantee the detection of the target. For pursuit evasion in a graph based environment, a worst case assumption is normally that the target can move along the edges with a arbitrary speed

[39]. In a continuous environment, the worst case target model can be a fixed speed recontamination model [43, 45]. The worst case assumption maybe be conservative, but it can be very efficient to approximate evasion effort of the targets.

The detailed assumptions and simplifications for the worst case model will be made in Section 5.4.

5.3 Construction of the Objective Functions

In the search and pursuit evasion game, not only the Objective function for the pursuer needs to be built, a sensible Objective function for the targets should also be formulated, for the pursuer to consider the possible rational behaviour of the targets. Assume that all the players have the same time horizon T for planning. Let a_t^λ be the action taken by target λ at time t. $a_t^\lambda = \pi^\lambda(s_t)$ denotes the policy of λ, deciding which action it takes given the system state. $\delta = \{\pi^\lambda(s_t) : \lambda \in \Lambda\}$ is the set of policies of all the targets. The pursuer action and policy, $a_t^p = \pi^p(\hat{s}_t)$, are defined respectively.

5.3.1 Objective Function of the Pursuer

In Chapter 3, it has introduced the idea of combining search and monitoring by building a united Objective function for the agent. The *uncertainty* is defined for the evasive targets. Let $\tilde{E}_\lambda(\hat{s}_t)$ be the *uncertainty* about target λ at subject state $\hat{s}_t$. Such *uncertainty* directly describes all the reachable locations of a target, thus providing a practical information for the estimation and prediction. For state $\hat{s}_t$, let $R(\hat{s}_t) = -\sum_{\lambda\in\Lambda}\tilde{E}_\lambda(\hat{s}_t)$ be the reward of the pursuer. The rationale of this reward formulation is to represent the total *uncertainty* level of the target locations. The Objective function of the pursuer for time horizon $T = t_f - t_i$ is the average expected reward at each time step.

$$
\begin{aligned}
V_\rho(\delta, \pi^p, \hat{s}_{t_i}, t_f) &= E\{\frac{\Delta T}{T}\sum_{t=t_i}^{t_f} R(\hat{s}_t)\} = \frac{\Delta T}{T}\sum_{t=t_i}^{t_f} E\{R(\hat{s}_t)\} \\
&= \frac{\Delta T}{T}\sum_{t=t_i}^{t_f}\sum_{\hat{s}_t\in\hat{S}} p(\hat{s}_t|\delta, \pi^p, \hat{s}_{t_i})R(\hat{s}_t) \\
&= -\frac{\Delta T}{T}\sum_{t=t_i}^{t_f}\sum_{\hat{s}_t\in\hat{S}} p(\hat{s}_t|\delta, \pi^p, \hat{s}_{t_i})\sum_{\lambda\in\Lambda_t}\tilde{E}_\lambda(\hat{s}_t)
\end{aligned}
$$

where ΔT is the time step of system, and $\hat{S}$ is the subjective state space of the agent.

As defined in previous section, the uncertainty of the targets develops with time, because of the target evasion. Thus the objective value can be increased by searching for uncertain targets, as well as revisiting and monitoring targets with low *uncertainty*. In such way, the search and monitoring is combined dynamically. A policy should be planned to allow the agent to do both missions simultaneously for the same goal.

5.3.2 Objective Function of the Targets

For the evasion, targets are assumed to have a short term interest of evading the upcoming detection. Assume the targets are selfish, that an evader only consider the detection time of itself. For target λ, at the initial time t_i, $tp^{\lambda}_{t_i}$ denotes the previous time instant when it was detected, where $tp^{\lambda}_{t_i} < t_i$. $tn^{\lambda}_{t_i}$ denotes the next time instant to be detected, where $tn^{\lambda}_{t_i} \geq t_i$. $p(tn^{\lambda}_{t_i}|\delta, \pi^p, s_{t_i})$ denotes the probability distribution of $tn^{\lambda}_{t_i}$, given the agent policy π^p, the set of target policies δ, and the initial state s_{t_i}.

Then the Objective function for target λ is formulated as its expected time of being detected.

$$V_{\lambda}(\delta, \pi^p, s_{t_i}, t_f) = E\{(tn^{\lambda}_{t_i} - tp^{\lambda}_{t_i})\} = \sum_{t=t_i}^{t_f} p(t|\delta, \pi^p, s_{t_i})(t - tp^{\lambda}_{t_i}) \tag{5.1}$$

where t_f denotes the end of the time horizon.

Note that this Objective function only considers the next immediate detection. Thus in application, each target should replan its policy after being detected or after escaping detection. If target λ is under measurement at time t, let $tn^{\lambda}_{t_i} = t$. Then any policy of λ has equal Objective value. Then while being sensed by the pursuer, assume that the target take policy π^{λ}_s, which makes the target stay unmoved. The rationale behind π^{λ}_s is that, when a target is under measurement, its effort to get out of the sensor footprint is trivial, because it can be easily outrun. The evader runs away only when the pursuer leaves it to find another target.

5.4 Simplification of Strategy Planning

It has been mentioned in Section 5.2 that the worst case assumption should be made on the targets. However, this work does not apply a certain assumption of the target

motion directly. Instead, an assumption is made on the information available to both sides of players. In Chapter 3, it has defined the sensing range of the pursuer and the targets, and we can see that this is a game with asymmetric information for both sides. In such games, the available information for a player is vital to its decision making. Therefore, by making an intermediate assumption on the information, then deducing how it simplifies the target motion models, the internal mechanics of the target behaviour can be analysed more explicitly, and design more sensible pursuit policy. Also it can make it easier to evaluate the conditions for such assumption to be valid, thus deciding the specific scenarios for the assumption to be applicable.

The following assumption is made to facilitate the simplification.

1. assume the time horizon T for planning is long enough, so that it is considered to be infinite in the later induction

In [30], an assumption was made that all the information available to the pursuer can be accessed by the targets. In this work, a further simplification is made that a target can not only access the locational information of the pursuer and other targets, but also have *knowledge and foresight* of the policy of the pursuer.

Definition 5.1. A target has *knowledge and foresight* of the pursuer policy $\pi^p(\hat{s}_t)$, if it can predict the output of the policy $a_t^p = \pi^p(\hat{s}_t)$ exactly, given the subjective state $\hat{s}_t$ of the pursuer, no matter whether π^p is a pure or mixed strategy.

The assumption of *knowledge and foresight* resembles the scenario that the pursuit policy is hacked or conjectured by the evaders. it will then be proved that this transforms the pursuit-evasion game into a recontamination problem.

Lemma 5.2. *Assume that every target knows the location of all other players, and has knowledge and foresight of the pursuer policy. Given the target Objective function defined in Section 5.3.2, and given a pursuer policy, the targets will plan a set of pure policies to reaches a Nash Equilibrium.*

Proof. At time t, each target knows the system state s_t exactly, because of their full observation. The measurement of the pursuer is dependent on the system state s_t, thus the targets can also precisely access the current subjective state $\hat{s}_t$. Let Λ_t^o be the set of targets which are being measured at time t, and $\Lambda_t^e = \Lambda - \Lambda_t^o$ is the set of unmeasured targets. When an evader is detected by the pursuer, or when an evader escapes from the sensor footprint, it is called an *incident*.

At initial time t_i, the measured targets $\lambda \in \Lambda^o_{t_i}$ take a known policy $\pi_s(s_{t_i})$, as mentioned in Section 5.3.2. With the *knowledge and foresight* of policy $\pi^p(\hat{s}_{t_i})$, the next location of the pursuer and the measured targets, which are $x_\rho(t_i+1)$ and $\{x_\lambda(t_i+1) : \lambda \in \Lambda^o_{t_i+1}\}$, can be predicted by the targets. If no *incident* happens at t_i+1, the observation of the pursuer y_{t_i+1} can also be predicted, and so is $\hat{s}_{t_i+1}$. Thus, if ignoring the occurrence of the next *incident*, a trajectory of the pursuer with indefinite length, $\chi_\rho(t_i,\infty) = \{x_\rho(t) : t = t_i, t_i+1, ...\}$, can be fully predicted deterministically by the targets iteratively.

At t_i, for each unmeasured target $\lambda \in \Lambda^e_{t_i}$, given $\chi_\rho(t_i,\infty)$, there is a fixed evasion path $\chi_\lambda(t_i)$, which can guarantee the target to be detected as late as possible. $\chi_\lambda(t_i)$ can be known by all other targets, because of the shared knowledge of $\chi_\rho(t_i,\infty)$. Given $\chi_\rho(t_i,\infty)$, π_s, and $\{\chi_\lambda(t_i) : \lambda \in \Lambda^e_{t_i}\}$, the first incident to happen and the happening time t_1 can be predicted. For the same reason as above, if not considering the next *incident*, the segment of pursuit trajectory, $\chi_\rho(t_1,\infty) = \{x_\rho(t) : t = t_1, t_1+1, ...\}$, can be predicted by all targets.

For an unmeasured target at t_1, $\lambda \in \Lambda^e_{t_1}$, if it has not been detected before t_1, it can re-plan its evasive path according to $\{\chi_\rho(t_i,t_1), \chi_\rho(t_1,\infty)\}$. For those which escape measurement at t_1, they plan the evasive path according to $\chi_\rho(t_1,\infty)$. Then the next *incident* can be predicted again.

Repeat this process iteratively for each target λ, until λ is supposed to be detected in the kth *incident*. We can see that, every target plans its evasive strategy with perfect foresight of the future trajectory of the pursuer, which is $\{\chi_\rho(t_i,t_1), \chi_\rho(t_1,t_2), ..., \chi_\rho(t_{k-1},\infty)\}$. Thus the evasive strategy is a fixed path, and it should be optimal so that it guarantees no other policy can make the target detected at later time. So it has been proven that, a Nash Equilibrium between all targets has been achieved by a set of pure policies that no player can improve its reward by unilaterally changing its policy.

□

Definition 5.3. The occupancy grid map of a target λ is *pushed to clear*, if the pursuer clears the map incrementally, until the last positive piece of map is covered. It is equivalent to that the target is detected only when there is no other possible locations for it to be at.

Theorem 5.4. *Assume that every target knows the location of all other players, and has knowledge and foresight of the pursuer policy. Given the Objective function defined in Section 5.3.2, the targets will plan a set of policies δ, so that the pursuer will detect a target λ only after it pushes to clear the occupancy grid map of λ.*

Proof. It is shown in the proof of Lemma 5.2, that each target plans its policy independently, with the perfect foresight of the future trajectory χ_ρ of the pursuer. For a target λ, at the moment it escapes measurement from the pursuer, before it being detected again, the occupancy grid map $\hat{\mathrm{M}}_\lambda(c, t|Y_t)$ can be predicted iteratively based on χ_ρ, as follows:

$$\hat{\mathrm{M}}_\lambda(c, t+1|Y_t) = sign(\sum_{c' \in \varsigma} \theta(c|c', t+1)\hat{\mathrm{M}}_\lambda(c', t|Y_t))$$
$$\forall c \in \Delta, \hat{\mathrm{M}}_\lambda(c, t+1|Y_{t+1}) = 0$$

where Δ is the sensor footprint, centred at the agent location at each time step.

Thus it can be deterministically predicted, about the evolution of the map before the target being detected. Then, there exist a fixed path χ_λ for target λ, to let it stay within the area where $\{c : \hat{\mathrm{M}}_\lambda(c, t|Y_t) = 1, c \notin \Delta\}$. Thus it can guarantee that the target λ will be detected only after the occupancy grid map being *pushed to clear*. This induction applies to every targets independently, thus the theorem is proved.

□

Lemma 5.2 and Theorem 5.4 shows that although each target has a selfish goal and has no cooperation, but with the assumption of *knowledge and foresight*, it can be guaranteed that the behaviour of every target is always the worst case for the pursuer.

5.5 Policy Planning with Simplified Target Model

Theorem 5.4 and its proof show that, with the information available to the targets, a target λ will not be detected before the pursuer *pushes to clear* its occupancy grid map. They also show that, the evolution of an occupancy grid map is deterministic with respect to the future trajectory of the pursuer. Then we can see that, for the pursuer, given a trajectory χ_ρ, the reward at each time step, which is based on the occupancy grid maps, is predictable. Hence, there is a direct mapping from a fixed trajectory χ_ρ of the pursuer, the initial state, and the terminal time instant, to its Objective value:

$$V_\rho(\delta, \pi_\rho, \hat{s}_{t_i}, t_f) = f(\chi_\rho, \hat{s}_{t_i}, t_f) \tag{5.2}$$

Thus we can see that, the policy planning is simplified to be a path planning. It is a dynamic guarantee search problem, where a fixed path is designed for the SSM, considering the worst case scenarios of the target behaviour. The path planning algorithm designed in Section 4.4 can be applied, to conduct such path planning problem. Once the pursuer designed its fixed trajectory, it will be known by all the targets, and they design their own paths to evade detection.

5.6 Simulation Evaluation and Validation

5.6.1 Case Study

There is a $80m \times 80m$ square environment ς, which is discretized into 20×20 cells. The agent sensor can cover 5×5 cells. There are 5 targets and 1 pursuer. The pursuer flies with speed $V_\rho = 20m/s$, and the maximum target speed is $V_t = 2.5m/s$. The agent will plan and execute a path χ_ρ with a planning horizon $T = 10s$. The targets always have access to χ_ρ, and design their evasive path accordingly. Different to Section 5.3.2, in simulation, the target will try to get out of sensor footprint of the agent as soon as possible when being measured. This is to make the simulation more realistic. The pursuer re-plan its policy when an unpredicted detection or evasion happens, or when the planning horizon is reached. The targets are randomly scattered initially, but initial locations are known to the pursuer. Figure 5.1 to 5.5 are the snapshots of the simulation, where the solid lines are the plan or actual trajectory of the pursuer, the dash lines are the trajectories of the targets.

Figure 5.1 shows the plan of the pursuer at the initial time instant. When the agent has perfect belief of all targets, the agent tries to cover each target in an efficient way to prevent the *uncertainty* to grow, which is similar to a Travelling Salesman Problem. Figure 5.2 shows the evolution of this pursuit evasion game from the initial time step, with the pursuit plan shown in Figure 5.1. The player location and occupancy grid map at different time instants are depicted, by adding a vertical axis of relative time instant. It shows that the agent can detect target 2, 5, and 1 along the route. The target 3 and 4 are not detected, but parts of their occupancy grid maps are cleared during this period. Thus the development of the total *uncertainty* is controlled in such a way.

Figure 5.3 shows the pursuer planning in the middle of the game. With more uncertainty on the far-away target 3, and with target 2 being sensed, the agent spends more effort to cover target 1,4 and 5, and tries to reach target 3 in the end. Figure 5.4 shows that target 1 was encircled and finally caught. Figure 5.5 shows that the agent cleared a big

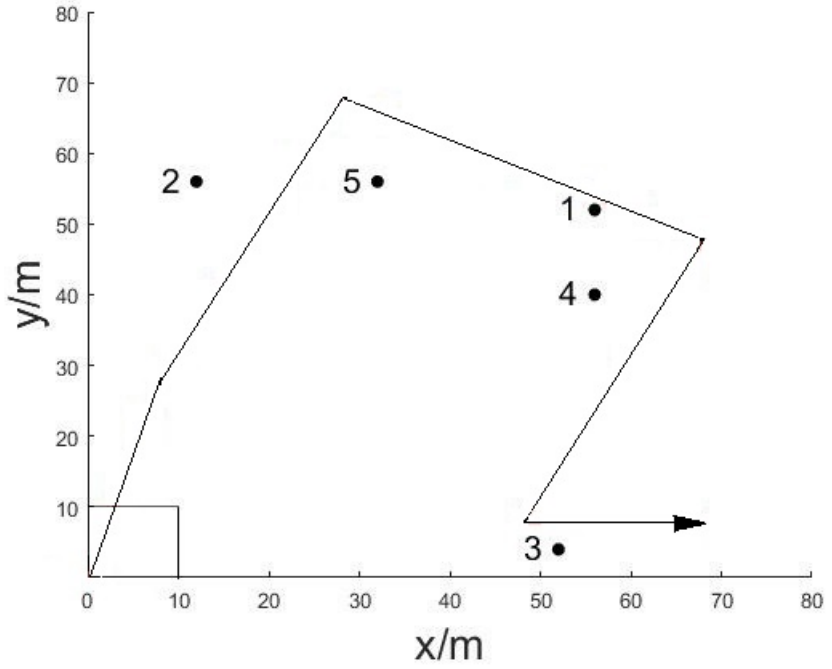


FIGURE 5.1: initial planning of the pursuer

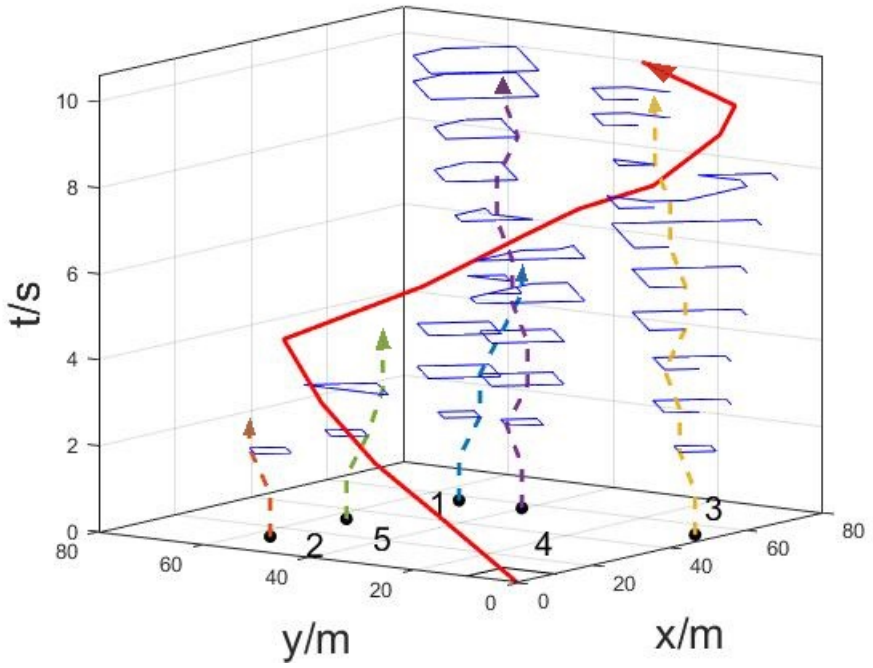


FIGURE 5.2: initial game playing

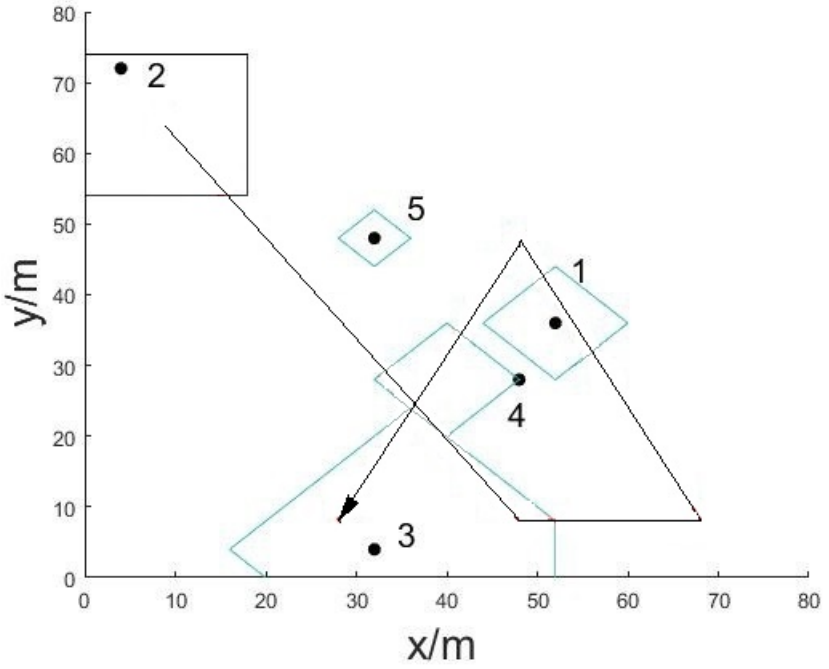


FIGURE 5.3: planning of the pursuer during the middle of the game

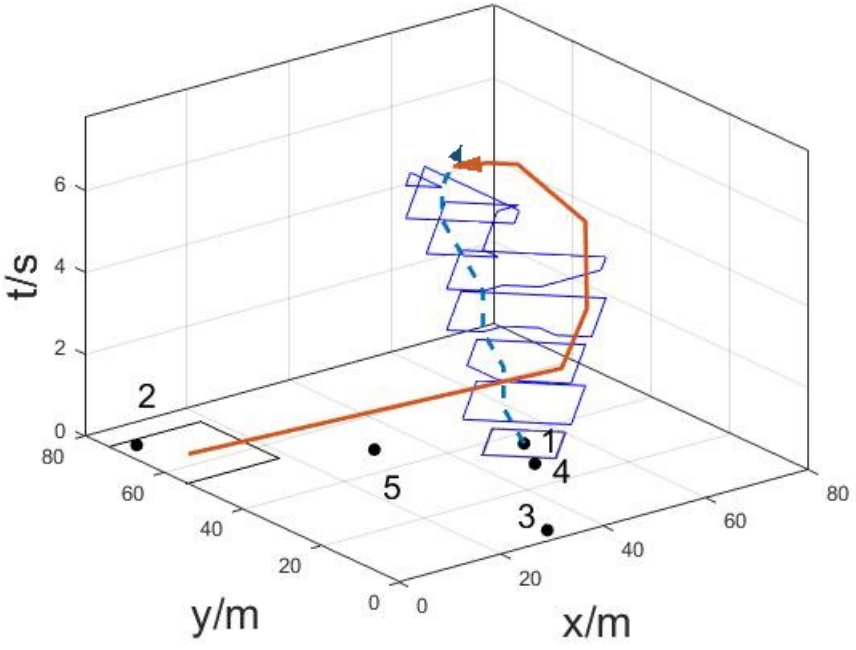


FIGURE 5.4: pursuit-evasion of target 1

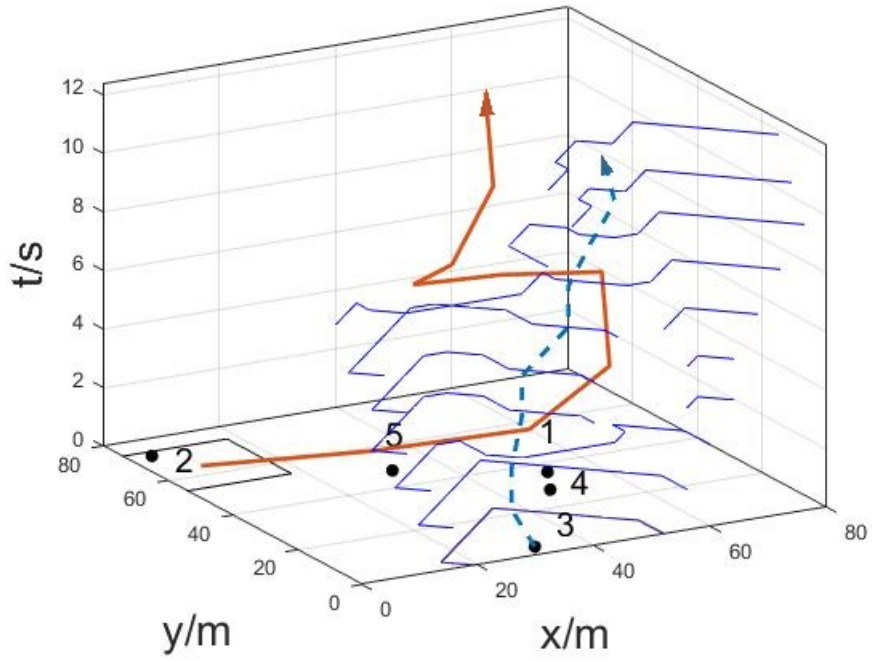


FIGURE 5.5: pursuit-evasion of target 3

part of the occupancy grid map of target 3, but target 3 still evaded to its reachable area and avoided detection.

The snap shots show that, with perfect foresight of the pursuer actions, the targets can always find a track to hind within its reachable area to evade detection. For the pursuer to find a target, it should *push to clear* the occupancy grid map of an evader, thus pushing it to its last reachable point.

Figure 5.6 shows the development of the *uncertainty* in a case study. We can see that, the pursuer can detect every target and clear the *uncertainty* to zero, for multiple times during the game. All of the targets can have their *uncertainties* maintained in a relatively low level for most of the times. The total *uncertainty* is thus maintained bellow a reasonable level. This shows that, even with the worst case assumption on the evaders, the agent is still capable of reducing and limiting the total *uncertainties* on the targets.

5.6.2 Comparative Study

To study the efficiency of the proposed policy planning, this approach is compared with the method in [42]. [42] studied a similar guaranteed search problem, and the pursuer takes a fixed pattern to sweep the whole rectangle area back and forth in parallel. There

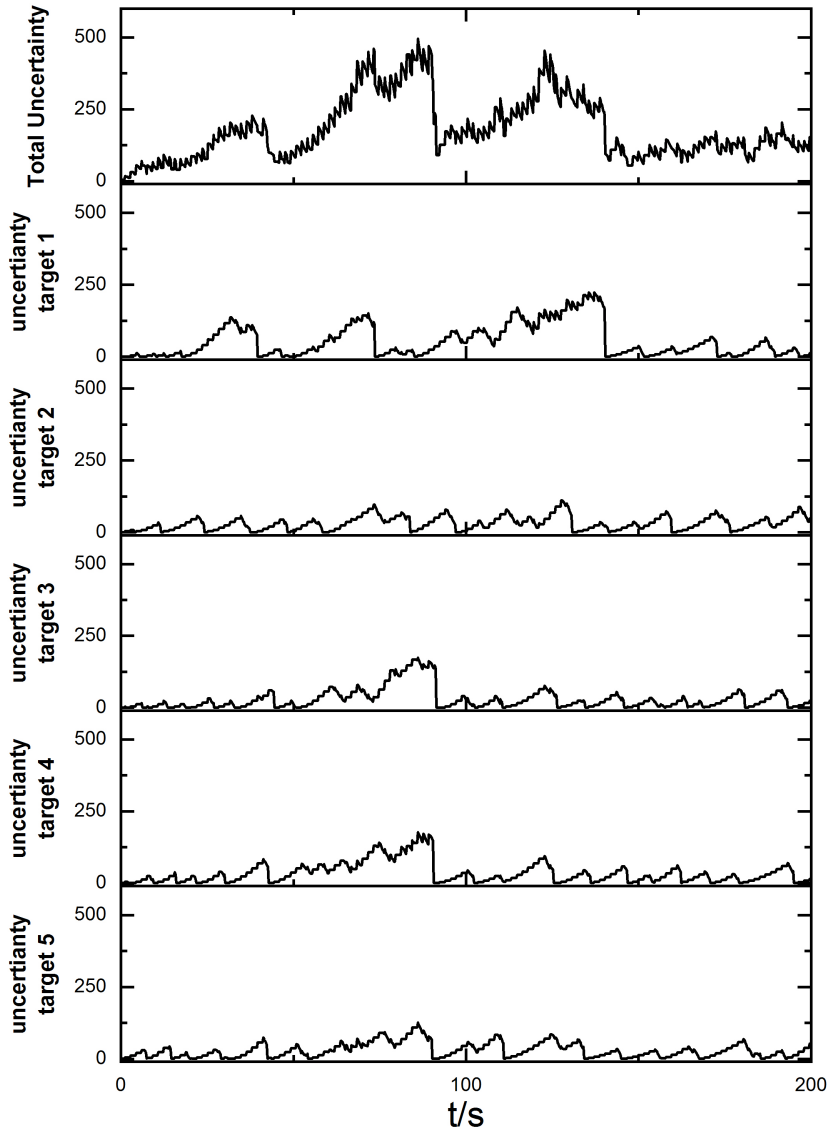


FIGURE 5.6: *uncertainty* reduction

is overlapping between each line of sweep to avoid recontamination. It is is a fixed-pattern guaranteed search approach. Such method is illustrated in Figure 5.7.

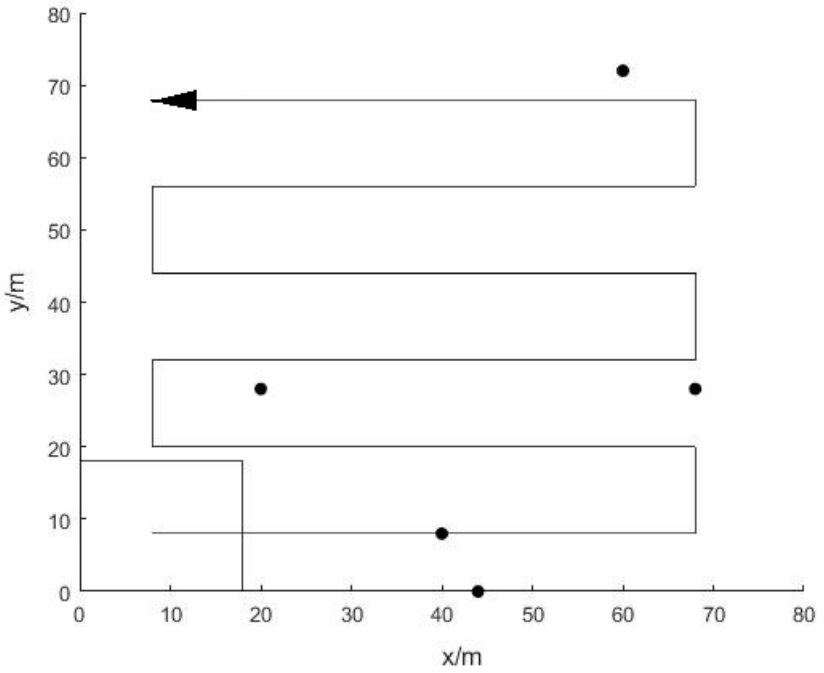


FIGURE 5.7: fixed pattern guaranteed search

Scenarios have been studied with n=2, 3, 5 and 7 targets, and with $V_t = 1m/s$, $1.5m/s$, $2m/s$, $2.5m/s$, $3m/s$, and $3.5m/s$. For each scenario, 100 cases of simulation are conducted, for 200 seconds long each. The total *uncertainty* in a scenario is the average total *uncertainty* at each time step of each cases, and the average computation time of each planning is recorded. Figure 5.8 illustrates the performances in each scenarios. Each case is done by one core of E5 2650V2 processor (2.6 GHz).

When $V_t \geq 3m/s$, there is no data for the fixed-pattern guaranteed search, because it is not feasible in these scenarios. But Figure 5.8 can still show that, in every scenario, the performance of the proposed policy planning can have significant advantage over the fixed pattern sweeping. It validates that, when the targets are evasive, by estimating possible evader actions according to current information, and planning the policy dynamically, the agent can achieve a better performance than a fixed pattern guaranteed search.

The computation time for the proposed dynamic policy planning is in Table 5.1. The computation time of the fixed pattern guaranteed search is not presented, because it is assumed that the pattern generation is instantaneous.

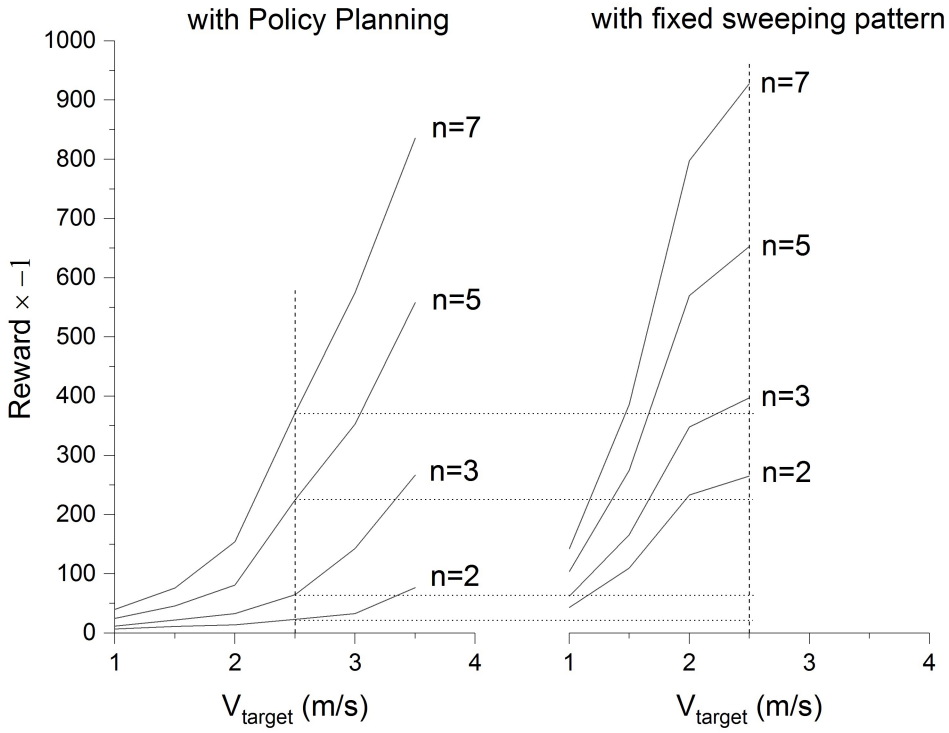


FIGURE 5.8: comparison of performances. solid lines are the performances of the proposed policy planning, and the dash lines are these of the fixed-pattern search

number of targets	2	3	5	7
planning time	0.20s	0.30s	0.47s	0.65s

TABLE 5.1: computation time for dynamic policy planning

The policy planning takes 0.41 second in average for each planning, which is promising to be applied in practice. Thus the proposed policy planning is chosen to be the solution of the single pursuer SSM of the evasive targets.

5.6.3 Considering the Manoeuvrability of the Pursuer

The same as in the SSM of randomly moving targets, the comparative study is done again while considering a minimum turning radius $r_c = 5m$ for the agent. The result is shown in Figure 5.9.

We can see that the advantage of the proposed policy planning is still very clear against the fixed pattern search. This proves that even in a realistic situation, the proposed dynamic guaranteed search can still efficiently search and Monitor multiple evasive targets.

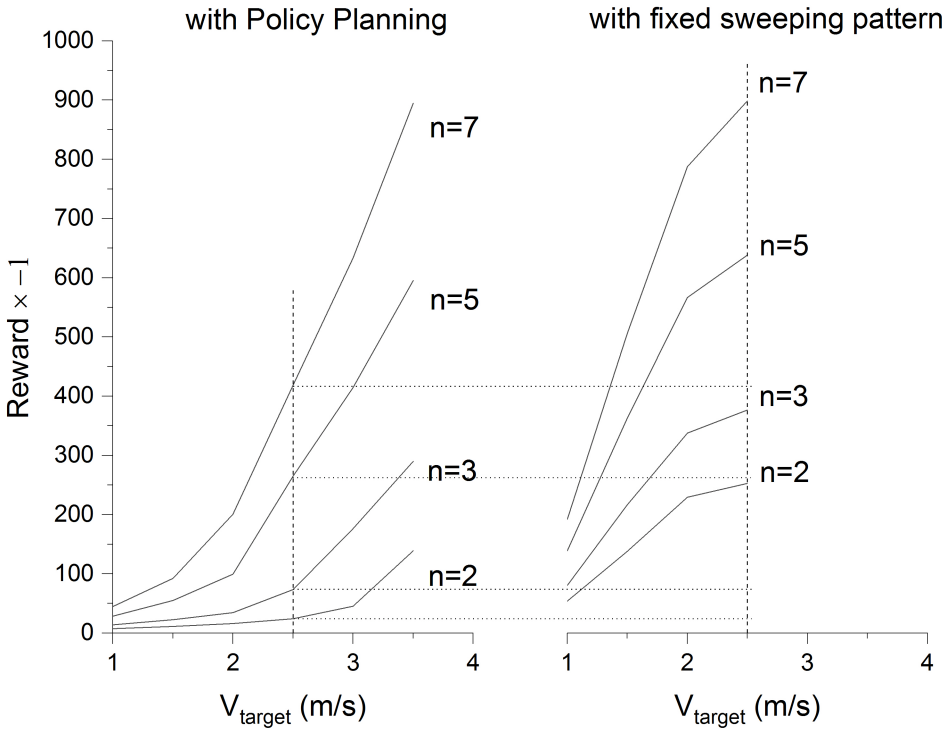


FIGURE 5.9: comparison of performances, with $r_c = 5m$. Solid lines are the performances of the proposed policy planning, and the dash lines are these of the fixed-pattern search

5.6.4 Exploring the limitations of SSM

The same as in section 4.5.4, in this section, the simulations in section 5.6.2 are expanded to some more extreme situations, to find out the practical limitations on SSM. We also expand the scope of simulation in two dimensions separately: the maximum velocity of targets and the size of environment. Compared with the simulation in Section 5.6.2, we either expand the V_t to $6m/s$ and $10m/s$, or expand the arena to $140m \times 140m$ and $180m \times 180m$. The results are shown in Figure 5.10 and 5.11:

For the guaranteed search with fixed pattern, it is not feasible when the width of environment $L = 180m$, and it is only feasible when $L = 140m$ and $V_t = 1m/s$. It can be seen from Figure 5.10 and 5.11 that the proposed dynamic guaranteed search still has significant advantage compared with fixed-patter guaranteed search. However, it shows the same tendency as in section 4.5.4 that, the faster speed of target evasion and bigger environment make SSM much more difficult for the agent. In Figure 5.10, when $V_t > 5m/s$, the uncertainty starts to grow out of control. When the target speed is considerably fast, because of their advantage in detecting the location of the agent, they can easily evade detection. In figure 5.11, we can also see that, when the width of

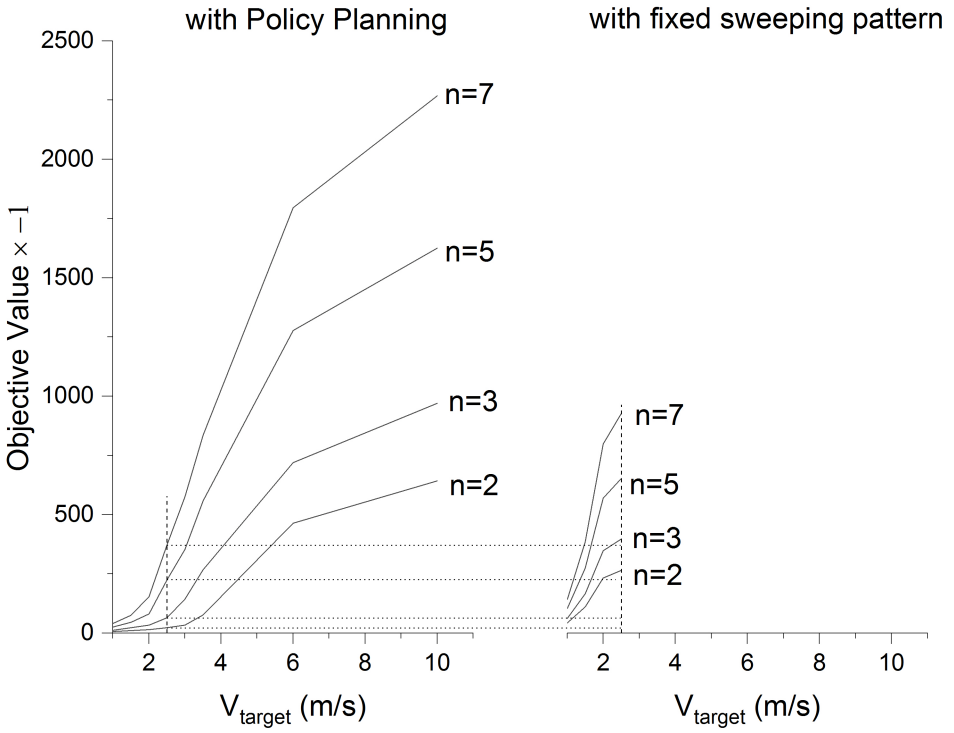


FIGURE 5.10: comparison of performances with environment size equals $100m \times 100m$, and with expanded range of V_t

environment $L \geq 180m$, the total uncertainty soon grows to more than 2000 even with $V_t = 3.5m/s$. In a big environment, the agent can not reach every possible region within suitable time interval, thus the uncertainty level can not be maintained. Therefore, when $V_t > 5m/s$ or when the environment is $180m \times 180m$, the agent is not capable of updating the locations of targets and keeping a low uncertainty level, which are the limitations of SSM with proposed dynamic guaranteed search.

5.7 Conclusion

For the single pursuer SSM of evasive targets, using POSG framework can be intractable, and the conventional heuristic methods are deemed not suitable by this work. In this Chapter, by making certain assumption on the information available to the targets, the difficult pursuit evasion game has been simplified to be a dynamic guaranteed search problem. The policy planning is transformed to be a path planning, which is more computationally feasible. The simulation shows that, with limited sensing range and limited speed, the agent can still efficiently search and monitor all the evasive targets simultaneously, to reduce the total *uncertainty* level. The comparative study proves

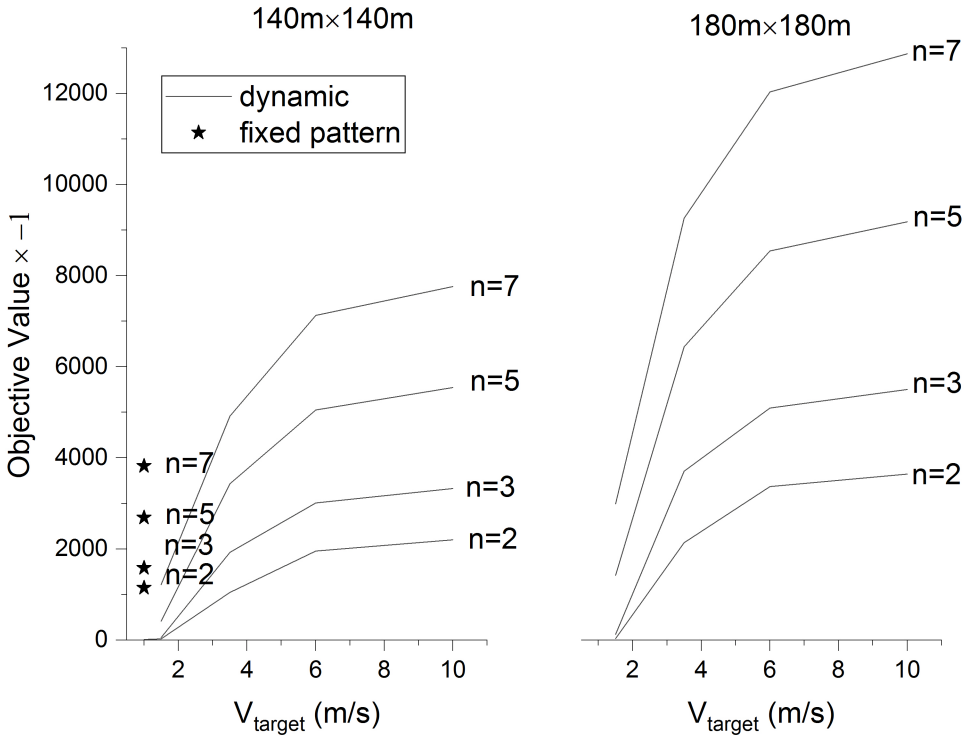


FIGURE 5.11: performance of SSM with environment size equals $140m \times 140m$ and $180m \times 180m$

that it has better performance over a guaranteed search method with a fixed sweeping pattern. In the realistic simulation when there is a minimum turning radius of the agents, the above conclusion still stands, and then the proposed policy planning is chosen to be the solution for this Chapter.

Although a worst case assumption is applied on targets, it is not directly on the motion model of the targets, but on the information available to the targets. This can be a theoretical analysis framework, to analyse the simplifications in other works.

The limitations of SSM of evasive targets are studied in Section 5.6.4. The agent will not be capable of performing SSM, when the target evasion speed or the size of environment is beyond a certain limit. However, as shown in Section 5.6.1 and 5.6.2, under moderate conditions, the SSM between a single agent and multiple evasive targets is still practical, and the above mentioned advantages of dynamic guaranteed search still hold.

Chapter 6

Cooperative Simultaneous Search and Monitoring

The single pursuer SSM of randomly moving targets and evasive targets has been studied in Chapter 4 and 5. In this chapter, the single pursuer SSM is extended to multiagent SSM. Both case with randomly moving targets and evasive targets are considered. By applying game theoretical methods, the agents can plan a joint set of policies in a distributed way, which can achieve cooperative SSM. Some background of cooperative strategy planning is given in Section 6.1. The cooperative policy plannings are designed for randomly moving targets and evasive targets separately, in Section 6.2 and 6.3.

For the case with randomly moving targets, it is very difficult to analyse and design the reactions of an agent to the observations and actions of other agents. Therefore, the concept of Partial Open-Loop Feedback Control is applied, to allow the local policy to react to only local information. This is combined with the heuristic reactive policy developed in Chapter 4, to have an intuitive design of cooperative strategy. For the case with evasive targets, the assumptions made in Chapter 5 are utilized to develop the cooperative strategy planning. Both the simulation and theoretical proof shows that, even though the local policy is planned in a distributed way and is reactive to only local observations, the cooperative SSM is still achieved, which has clear advantage over non-cooperative SSM.

To test the practicability of the cooperative SSM in realistic applications, the communication range and the minimum turning radius are considered in Section 6.5 and 6.6. The simulation results support all the major conclusions that has been drawn by studying with the ideal models. The inclusion of communication range also improve the scalability of the cooperative SSM.

Please note that collision avoidance is not considered in this work.

6.1 Distributed Online Strategy Planning

The SSM is an information gathering problem, thus the benefit of measurement on a certain region or a target will saturate, with excessive effort being spent on it. It is not advisable to let each agent implement the single pursuer SSM on its own regardless of each other, which may cause undesired overlap of efforts. One way of exploiting the advantage of multiple agents is to partition the environment and the targets, and let each agent focus on a certain part. Some works use heuristic method to divide the environment. They utilize the target and agent information, to approximate the possible reward of a certain partition [11, 12]. The task allocation method takes similar idea, by abstracting the goal into separate tasks, and assign them to each agent according to the best task scheduling [3, 23, 29]. In this work, when the targets presence and motion are uncertain, the heuristic partitioning or abstraction can not guarantee a precise approximation of the actual performance. A coordination scheme with look-ahead ability is needed.

For the Multiagent search and pursuit evasion problem with no communication or limited communication, the non-myopic solution can be formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [68], or a Partially Observable Stochastic Game (POSG) [69]. These problems can be solved online [69] or offline [68]. These methods can be rigorous, but are computationally difficult [32]. For the same reason as discussed in Section 4.3.1, the offline methods can be intractable and non-adaptive to the environmental changes. Therefore this work chooses online solution methods for the cooperative strategy planning, to have a feasible and adaptive application.

The real-time information sharing between the pursuers has been assumed in Section 3.2.3, thus the agents do not need to estimate the observations of each other, which largely simplifies the problem. For some works [65, 66], the perfect communication is also assumed, and the strategy planning is centralized, which is a Multiagent Partial Observable Markov Decision Process (MPOMDP). The joint policies are planned for all agents offline, under the assumption that the agents have full online communication for their measurement. However, a joint policy contains more information compare with the information of detection, thus a centralized online policy planning will demand a high bandwidth communication, and is less feasible and reliable in non-ideal environment. Thus in this work, a distributed policy planning approach will be developed. Each agent plans a strategy in a distributed way for a common time horizon, and joint policy should achieve the cooperation between agents.

6.2 Multiagent Simultaneous Search and Monitoring of Randomly Moving Targets

6.2.1 Formulating the Decentralized Partially Observable Markov Decision Process

A POMDP has been built for the single pursuer SSM of randomly moving targets. With multiple pursuer, the POMDP built for single pursuer SSM is extended into a Dec-POMDP, which can be described in a tuple $\langle I, S, \{A_i : i \in I\}, \{\Omega_i : i \in I\}, T, \{O_i : i \in I\}, \{R_i : i \in I\}\rangle$ [87], where

1. Γ is a finite set of pursuers. $\rho \in \Gamma$ is the label of a certain pursuer.
2. S is a finite set of states of the world. At time t, a belief state $s_t = \{\{\hat{\mathrm{P}}_\lambda(c,t|Y_t) : \lambda \in \Lambda\}, \{\hat{\mathrm{x}}_\lambda(t) : \lambda \in \Lambda_t\}, \{x_\rho(t) : \rho \in \Gamma\}, \Lambda_t, t\}$. $s_t \in S$;
3. A_s^ρ is a finite set of possible actions of the agent ρ at state s. $A_s = \{A_s^\rho : \rho \in \Gamma\}$ is a joint set of possible actions at state s. Let $a_\rho \in A_s^\rho$ denote the action of agent ρ. Let $\theta = \{a_\rho : \rho \in \Gamma\} \in A_s$ denote the joint actions of the agents;
4. $p(s_{t_+}|s_t, \theta) : S \times A_s \longrightarrow p(S)$ is the state-transition function, mapping from a previous belief state and the joint agent actions, to a probability distribution of next belief states;
5. $R : S \times A_s \longrightarrow R$ is the joint reward function, mapping from a current world state and the joint agent actions, to a immediate common reward;
6. Ω_ρ is a finite set of the observation of the agent ρ, where $y_t^\rho \in \Omega_\rho$ is the individual observation at time t;
7. $O_\rho : S \times A_s^\rho \longrightarrow p(\Omega_\rho)$ is the observation function, mapping from a current world state and an action of the agent ρ, to a probability distribution of the observations of the agent ρ.

The multi-agent SSM in this work is simpler than the general form Dec-POMDP, in that the individual observation y_t^ρ is shared among all the pursuers. Thus the joint observation $y_t = \{y_t^\rho : \rho \in \Gamma\}$ can be accessed by all the agents, and so does the common belief state s_t. This Dec-POMDP differs from the POSG built in Chapter 5, in that the Reward is also commonly shared among the agents, rather than self-interested or contradicting.

The state-transition function $p(s_{t_+}|s_t, \theta)$ takes the same definition as in the single pursuer scenario. It is based on the update rules of $\hat{\mathrm{P}}_\lambda(c, t|Y_t)$, $\hat{\mathrm{x}}_\lambda(t)$, and Λ_t, which are defined in Chapter 3.

The reward function $R(s_t) = \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(s_t)$ is also defined the same with in single pursuer SSM. Let π_ρ denote the policy of agent ρ and $\Pi = \{\pi_\rho : \rho \in \Gamma\}$ to be the set of joint policies, then the objective function in Section 4.2 becomes:

$$
\begin{aligned}
V(\Pi, s_{t_i}, t_f) &= E\{\frac{\Delta T}{T} \sum_{t=t_i}^{t_f} R(s_t)\} = \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} E\{R(s_t)\} \\
&= \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\Pi, s_{t_i}) R(s_t) \\
&= \frac{\Delta T}{T} \sum_{t=t_i}^{t_f} \sum_{s_t \in S} p(s_t|\Pi, s_{t_i}) \sum_{\lambda \in \Lambda_t} \tilde{B}_\lambda(s_t)
\end{aligned}
$$

Thus, the goal of each agent is to plan a policy π_ρ independently, given the current common belief state and future joint observations. The local policies constitute a set of joint policies Π, to achieve the optimal overall objective value $V^*(\Pi, s_{t_i}, t_f)$, where

$$
V^*(\Pi, s_{t_i}, t_f) = \max_{\Pi} V(\Pi, s_{t_i}, t_f) \tag{6.1}
$$

6.2.2 Cooperation based on Partial Open-Loop Feedback Control and Cooperative Equilibrium

It has been stated in Chapter 2 that, solving POMDP precisely is PSPACE-hard [72], and solving Dec-POMDP is NEXP-complete [32]. Thus, just like what has been done in the single pursuer SSM, an exact solution is not favoured, and approximations are applied to yield practical solution instead. In Chapter 4 and Section 4.3.3.4, the policy planning is simplified by defining some heuristic reaction to the measurements. In the multi-agent case, besides its own observation, each agent also needs to consider the measurements and actions of other agents, in order to achieve the cooperation. However, it is not easy to conceive an intuitive way of how an agent should react to the information from other agents. Therefore, the concept of Partial Open-Loop Feedback Control is innovatively applied [98]. Under such assumption, when doing policy planning, each agent takes the current joint belief state, and possible strategies of other agents into account. But the local policy will only react to its local observation, and will ignore the future actions

and measurements of other agents. By taking such assumption, in the policy planning, the coupling between the actions of each agent can be disentangled. The cooperation between pursuers is achieved by coordinating the future local strategy of each agents, given current common information. Let $\hat{\pi}_\rho(y_t^\rho)$ be the local policy of agent ρ, in which y_t^ρ denotes the local measurement of ρ. $\hat{\Pi} = \{\hat{\pi}_\rho(y_t^\rho) : \rho \in \Gamma\}$ is the set of joint policies of Partial Open-Loop Feedback Control.

For the solution of cooperative search and pursuit evasion, the key concept is to find a Cooperative Equilibrium among the strategies of all agents [68, 69], in which no agent can improve its reward by unilaterally change its strategy. In this case where the reward is commonly shared, the optimal solution should also be a Cooperative Equilibrium. In this work, each agent finds a best set of joint policies $\hat{\Pi}^*$ which can ensure the highest overall objective value, and implement its corresponding local policy. Then, for pursuer ρ, its policy planning becomes,

$$\begin{aligned} &\text{plan } \hat{\Pi}^* = \underset{\hat{\Pi}}{argmax} V(\hat{\Pi}, s_{t_i}, t_f) \\ &\text{execute } \hat{\pi}_\rho^*(y_t^\rho) \in \hat{\Pi}^* \end{aligned} \tag{6.2}$$

Because the information is shared among agents, the calculation of the joint policies should work symmetrically for every agent. It is ignored that multiple equilibrium solutions exist with the same objective value. Thus it is assumed that by having this distributed planning, an identical sub-optimal set of joint policies will be generated by each agent, and each agent will implement its corresponding local policy to achieve the cooperative SSM. The key concept is that, the agents do not communicate about the strategies to take, but by anticipating the most rational mutual behaviours, they will reach a consensus on the joint policies.

6.2.3 Heuristic Reactive Local Policy

By taking the assumption of Partial Open-Loop Feedback Control, it has been assumed that, when planning the policy, each agent does not consider the future observations or actions of other agents. Thus this work focus on achieving the cooperation by partitioning the environment and targets, and letting each agent attend different parts, rather than by the synergistic efforts on the same area or targets. Then the following assumption/simplification is made:

6. **No Overlapping in Reward**. The trajectories of the pursuers may cross, but the overlap of any effort and reward between agents is not considered.

The Assumption 6 shows that each agent handle different area and targets independently, which further simplifies the policy planning. So when doing the calculation in Equation (6.2), the local policies are not allowed to have overlapping attempts with each other.

In Section 4.3.3, it has been proven that, for a single agent policy which only react to the local information, it can be precisely reconstructed by a base trajectory and a branching rule. Then for the local policy $\hat{\pi}_\rho(y_t^\rho)$ which is of the same structure, it can also be fully reconstructed with a base trajectory χ_ρ and a rule of branching. In the single pursuer SSM, a heuristic reactive branching rule is defined in Section 4.3.3.4. It has been proven that, the objective value achieved by the heuristic reactive policy is better than the conventional policy of FSOA, and can be of practical computational efficiency. Therefore, in the multi-agent SSM, this work let the local policy $\hat{\pi}_\rho(y_t^\rho)$ apply the same structure as the heuristic reactive policy defined in Section 4.3.3.4, which is written as $\hat{\pi}_a^\rho(y_t^\rho, \chi_\rho, h_t)$. $\hat{\Pi}_a$ is the corresponding set of joint policies. It will then be proven that, in the multi-agent SSM scenario, having the heuristic reactive policy as the local policy is still better than implementing the policy of FSOA.

Theorem 6.1. *The optimal set of joint heuristic reactive policies* $\hat{\Pi}_a^* = argmax_{\hat{\Pi}_a} V(\hat{\Pi}_a, s_{t_i}, t_f)$ *has an better estimated objective value than that of optimal joint policies of FSOA* $\hat{\Pi}_f^*$.

Proof. According to Assumption 6, there is no overlap of the reward achieved by each pursuer. Let $V_\rho(\hat{\pi}^\rho, s_{t_i}, t_f)$ be the local objective value of the information gathering by agent ρ, regardless of the other pursuers. Then we can see that

$$V(\hat{\Pi}, s_{t_i}, t_f) = \sum_{\rho \in \Gamma} V_\rho(\hat{\pi}^\rho, s_{t_i}, t_f) \tag{6.3}$$

Let $\hat{\Pi}_f^* = argmax_{\hat{\Pi}_f} V(\hat{\Pi}_f, s_{t_i}, t_f)$ be the optimal set of joint policies of FSOA. Then,

$$V(\hat{\Pi}_f^*, s_{t_i}, t_f) = \sum_{\rho \in \Gamma} V_\rho(\hat{\pi}_f^{\rho*}, s_{t_i}, t_f) \tag{6.4}$$

where $\hat{\pi}_f^{\rho*} \in \hat{\Pi}_f^*$.

From Theorem 4.7, we can see that, for each agent ρ and the policy of FSOA $\hat{\pi}_f^{\rho*}$, there should always exist a heuristic reactive policy $\hat{\pi}_a^{\rho*}$, so that

$$V_\rho(\hat{\pi}_a^{\rho*}, s_{t_i}, t_f) \geq V_\rho(\hat{\pi}_f^{\rho*}, s_{t_i}, t_f) \tag{6.5}$$

Let $\hat{\Pi}_a^* = \{\hat{\pi}_a^{\rho*} : \rho \in \Gamma\}$ be the corresponding set of joint policies, then we get

$$V(\hat{\Pi}_a^*, s_{t_i}, t_f) = \sum_{\rho\in\Gamma} V_\rho(\hat{\pi}_a^{\rho*}, s_{t_i}, t_f) \geq \sum_{\rho\in\Gamma} V_\rho(\hat{\pi}_f^{\rho*}, s_{t_i}, t_f) = V(\hat{\Pi}_f^*, s_{t_i}, t_f) \tag{6.6}$$

it thus proves that the optimal set of joint heuristic reactive policies should have at least better gain than that of the optimal set of joint policies of FSOA.

□

Theorem 6.1 justifies the choice of heuristic reactive policy to be the local policy. Before designing the detailed planning of $\hat{\Pi}_a$, it first needs to be proven that, although the local policy does not react to the other agents, the cooperation is achieved advantageously compared with having no cooperation. Let $\hat{\Pi}_{as}$ be the set of joint non-cooperative heuristic reactive policy, if each agent plans its self-interested local heuristic reactive policy $\hat{\pi}_{as}^\rho = argmax_{\hat{\pi}_a^\rho} V_\rho(\hat{\pi}_a^\rho, s_{t_i}, t_f)$, in the same way as in the single pursuer SSM which does not consider any cooperation. The performance of cooperative and non-cooperative SSM is them compared.

Theorem 6.2. *For the optimal joint policies $\hat{\Pi}_a^*$ obtained through Equation (6.2), it can achieve at least higher overall objective value than the non-cooperative joint policies $\hat{\Pi}_{as}$.*

Proof. For the local policies in $\hat{\Pi}_{as}$, there may be overlap between individual efforts. According to Assumption 6, the redundant part of the local objective values will not be counted repetitively, then

$$V(\hat{\Pi}_{as}, s_{t_i}, t_f) = \sum_{\rho\in\Gamma} V_\rho^{'}(\hat{\pi}_{as}^\rho, s_{t_i}, t_f) \tag{6.7}$$

where $V_\rho^{'}(\hat{\pi}_{as}^\rho, s_{t_i}, t_f)$ is the independent part of the local objective value of agent ρ, except for its redundant reward.

Considering the fact that the SSM is a sensor scheduling problem of information gathering, and that the effort of each agent may be partly overlapped with the others when applying $\hat{\pi}_{as}^\rho$, for each agent, there should exist a policy $\hat{\pi}_{as}^{\rho'}$, which only focuses on the

independent part of the attempts made by the original policy $\hat{\pi}^{\rho}_{as}$. Because of such concentration of effort, we can see that

$$V_{\rho}(\hat{\pi}^{\rho'}_{as}, s_{t_i}, t_f) \geq V'_{\rho}(\hat{\pi}^{\rho}_{as}, s_{t_i}, t_f) \tag{6.8}$$

Let $\hat{\Pi}'_{as} = \{\hat{\pi}^{\rho'}_{as} : \rho \in \Gamma\}$ be the set of such non-overlapping policies. Equation 6.2, together with Assumption 6, imply that $\hat{\Pi}^{*}_{a}$ can achieve the highest overall objective value, among all sets of non-overlapping joint policies. Then

$$\begin{aligned} V(\hat{\Pi}^{*}_{a}, s_{t_i}, t_f) &= max_{\hat{\Pi}_a} V(\hat{\Pi}_a, s_{t_i}, t_f) \geq V(\hat{\Pi}'_{as}, s_{t_i}, t_f) \\ &\geq \sum_{\rho \in \Gamma} V'_{\rho}(\hat{\pi}^{\rho}_{as}, s_{t_i}, t_f) = V(\hat{\Pi}_{as}, s_{t_i}, t_f) \end{aligned} \tag{6.9}$$

Thus the Theorem has been proven. □

Theorem 6.2 demonstrates that, notwithstanding the fact that the cooperation is not considered in the local policies themselves, the cooperation can still be achieved with benefit, by avoiding redundancy in the search and monitoring efforts, and by coordinating these self-interested local strategies in the planning.

By letting the local policy have the structure of a base trajectory plus a heuristic reactive branching rule, the local policy $\pi_{\rho}(y^{\rho}_{t})$ can be represented by a local base trajectory χ_{ρ}. The Equation 6.2 can be rewritten as

$$\begin{aligned} \Psi^{*} &= \underset{\Psi}{argmax} V(\Psi, s_{t_i}, t_f) \\ \Psi &= \{\chi_{\rho} : \rho \in \Gamma\} \end{aligned} \tag{6.10}$$

where Ψ is the joint base trajectory of all agents.

$V(\Psi, s_{t_i}, t_f)$ can be estimated by Monte-Carlo Sampling in the same way as in single pursuer scenario in Section 4.3.4. Thus a mapping from a joint base trajectory Ψ to the overall objective value $V(\Psi, s_{t_i}, t_f)$ is obtained. Then the computation of the optimal joint polices has become planning an optimal joint base trajectory Ψ^{*}.

6.2.4 Joint Path Planning

In the single pursuer SSM scenario, a path planning algorithm based on Simulated Annealing is proposed in Section 4.4. It is composed of a candidate trajectory mutation function and a simulated annealing algorithm. Too apply this method on the planning of multiple paths, some adjustment on the candidate trajectory mutation function needs to be done, to let it generate joint trajectories as solutions.

The candidate trajectory mutation function $\hat{\chi} = M(\chi)$ is extended to be a joint trajectory mutation function $\hat{\Psi} = M(\Psi)$. $M(\Psi)$ includes the four independent mutations on an individual $\chi_\rho \in \Psi$, which are defined in Section 4.4.2. In each round, the function may choose one local base trajectory randomly, and apply one of the four independent mutations; or it may choose two local trajectories by chance, and apply the fifth mutation:

5. **Segment Swap:**. two base trajectories $\chi_{\rho_1}, \chi_{\rho_2} \in \Psi$ swap parts of their paths, which is shown in Figure 6.1.

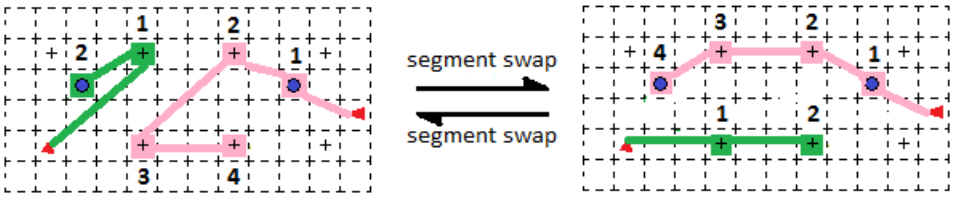


FIGURE 6.1: joint mutation on trajectories

The new joint trajectory mutation function can be used to generate different sets of joint policies, which can be incorporate into the Simulate Annealing algorithm defined in Section 4.4.3. an optimal joint base trajectory Ψ^* can be calculated iteratively.

Now the planning of local policy $\pi_\rho(y_t^\rho)$ is designed for each agent, which can be calculated and implemented in a distributed way.

6.3 Multiagent Simultaneous Search and Monitoring of Evasive Targets

6.3.1 Building the Partially Observable Game Playing

For the SSM between multiple pursuers and multiple evaders, it can be formulated as a Partially Observable Game Playing formulation, which is represented by a tuple $\langle I, S, \{A_i : i \in I\}, \{\Omega_i : i \in I\}, T, \{O_i : i \in I\}, \{R_i : i \in I\}\rangle$ [87], where

1. I is a finite set of players, which are the pursuers and the evaders. $i \in I$ is the label of a certain player.
2. S is a finite set of states of the world. The state $s_t = \{\{x_\rho(t) : \rho \in \Gamma\}\}, \{x_\lambda(t) : \lambda \in \Lambda\}$. $\hat{s}_t = \{\{\hat{\mathrm{M}}_\lambda(c, t|Y_t) : \lambda \in \Lambda\}, \{x_\rho(t) : \rho \in \Gamma\}\}$ is the subjective state of the pursuers;
3. A_i is a finite set of actions of the player i;
4. $T : S \times \{A_i : i \in I\} \longrightarrow p(S)$ is the state-transition function, mapping from a previous world state and the joint actions of all the players, to a probability distribution of next world states;
5. $R_i : S \times A_i \longrightarrow R_i$ is the reward function, mapping from a current world state and a player action to an immediate reward to that player;
6. Ω_i is a finite set of observation of the player i;
7. $O_i : S \times A_i \longrightarrow p(\Omega_i)$ is the observation function, mapping from a current world state and an action of the player i, to a probability distribution of the observations of the player i.

This differs to the single pursuer scenario in that the multiple pursuers are included in the set of players. With the same assumptions as in the single pursuer case, the state transition function T, the observation function O_i, and the Reward functions for the evaders $\{R_i : i \in \Lambda\}$ take the same definition as in Chapter 5. The agents have a shared objective function $V_p(\delta, \pi_p, s_{t_i}, t_f)$, where

$$V_p(\delta, \mathit{\Pi}, s_{t_i}, t_f) = E\{\frac{\Delta T}{T}\sum_{t=t_i}^{t_f} R(s_t)\} = \frac{\Delta T}{T}\sum_{t=t_i}^{t_f} E\{R(s_t)\}$$
$$= \frac{\Delta T}{T}\sum_{t=t_i}^{t_f}\sum_{s_t\in S} p(s_t|\delta, \mathit{\Pi}, s_{t_i})R(s_t)$$
$$= -\frac{\Delta T}{T}\sum_{t=t_i}^{t_f}\sum_{s_t\in S} p(s_t|\delta, \mathit{\Pi}, s_{t_i})\sum_{\lambda\in\Lambda_t}\tilde{E}_\lambda(s_t)$$

π^ρ and $\mathit{\Pi} = \{\pi^\rho : \rho \in \mathit{\Gamma}\}$ denote the local and joint policies of the pursuers. Then, the goal of each agent is to plan a policy π^ρ independently, given the current common belief state and joint observations. The local policies constitute a set of joint policies $\mathit{\Pi}$, to achieve the optimal overall objective value $V_p^*(\delta, \mathit{\Pi}, s_{t_i}, t_f)$, where

$$V_p^*(\delta, \mathit{\Pi}, \hat{s}_{t_i}, t_f) = \max_{\mathit{\Pi}} V_p(\delta, \mathit{\Pi}, \hat{s}_{t_i}, t_f) \tag{6.11}$$

6.3.2 Cooperative Policy Planning based on Simplified Target Model

In Chapter 5 Section 5.4, with the assumption of *knowledge and foresight*, it has already been deduced in Theorem 5.4 that, the Reward for the single pursuer develops deterministically, with respect to the sequence of actions of the pursuer. The same induction can be applied for the multiple pursuer scenarios, thus It can be said that, the joint Reward of the agents is directly mapped from the sequence of joint actions of the pursuers. Then, the joint set of policies of the pursuers, $\mathit{\Pi}$, can be simplified to be a joint set of fixed paths, $\mathit{\Psi}$. There exist a mapping, where

$$V_p(\delta, \mathit{\Pi}, \hat{s}_{t_i}, t_f) = f(\mathit{\Psi}, \hat{s}_{t_i}, t_f) \tag{6.12}$$

Because the pursuit policy is not reactive, the cooperative policy planning is simple. Let each agent find a best set of joint trajectories $\mathit{\Psi}^*$ which can ensure the highest overall objective value, and implement the corresponding local path. Thus for pursuer ρ, its policy planning becomes,

$$\begin{aligned} \mathit{\Psi}^* &= \underset{\hat{\mathit{\Psi}}}{argmax} V(\mathit{\Psi}, \hat{s}_{t_i}, t_f) \\ \chi_\rho^* &\in \mathit{\Psi}^* \end{aligned} \tag{6.13}$$

The same as in the SSM of randomly moving targets, the cases where multiple solutions exists with the same objective value are ignored. Thus such decentralized policy planning can guarantee the optimal global performance of SSM. Given the mapping from the joint trajectories to the objective value, the Algorithm of joint path planning in Section 6.2.4 can be applied to conduct the planning of optimal joint trajectories Ψ^*

6.4 Simulation Evaluation and Validation

6.4.1 Simulation of the Multiagent Simultaneous Search and Monitoring of Randomly Moving Targets

6.4.1.1 Case Study

The case study is done, with the same set-up of the environment in Section 4.5.1, and the same properties of the pursuers and the targets. But instead of having only one pursuer, there are 3 agents doing the SSM cooperatively. Each agent plans the optimal set of joint heuristic reactive policies $\hat{\Pi}_a^*$, and executes its local policy $\hat{\pi}_a^{\rho*} \in \hat{\Pi}_a^*$. The replanning is triggered at the same condition as in the single pursuer scenario, and it happens for all pursuers simultaneously.

Figure 6.2 and 6.3 are two snapshots of the case simulation.

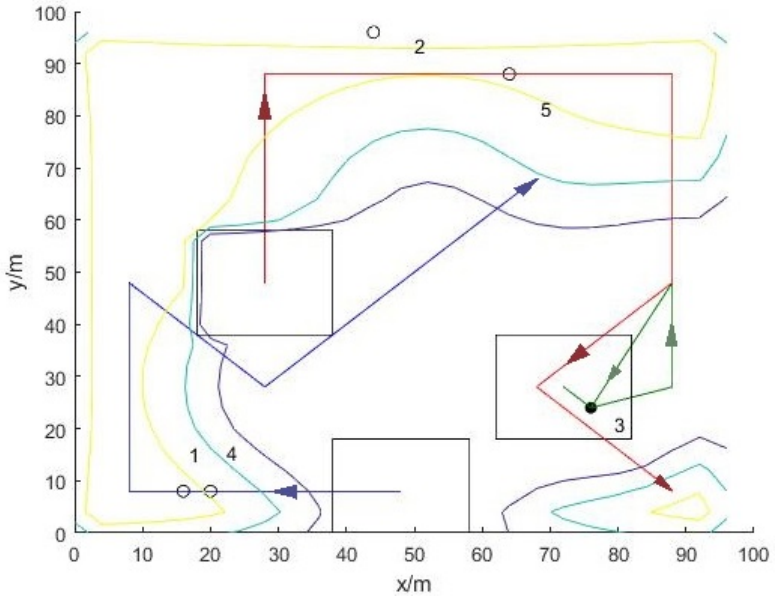


FIGURE 6.2: cooperative search

In Figure 6.2, there is only one target known. The agent which is currently covering the target chooses to focus on monitoring, but does some search in the nearby area in the

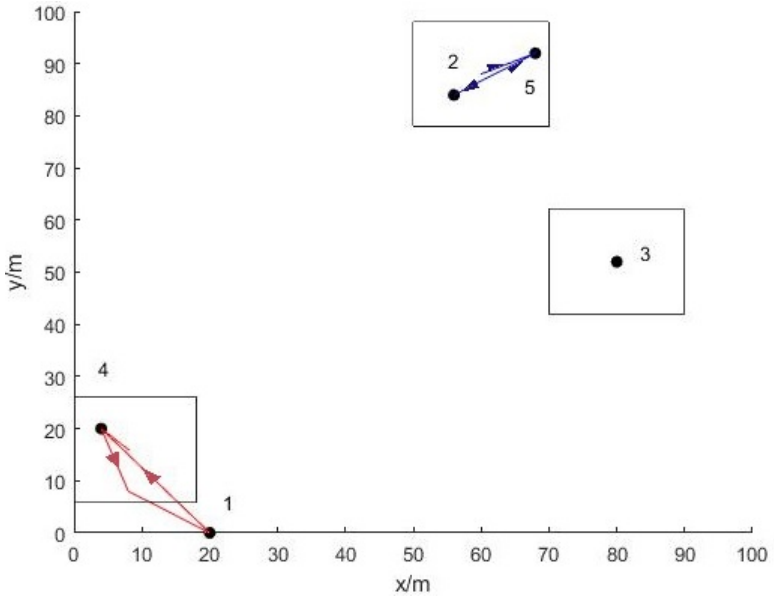


FIGURE 6.3: cooperative monitoring

mean time. The other two agents divide the unknown areas and plan non-overlapping trajectories for search. In Figure 6.3, when all the evaders has been detected, each agent goes back and forth to monitor nearby targets, and there is no redundant effort between the pursuers. Note that the cooperation of SSM is achieved in a distributed way, without any communication about plans. The case study shows that, with exchanging only sensing data, the agents can partition the tasks without overlap, and do the SSM with synergy.

Figure 6.4 shows the development of the *belief probability* of each target and the overall reward of the SSM mission, in a case study. We can see that, all targets can be detected soon after the beginning, and can be maintained a high *belief probability* for most of the times. Some sporadic negative spikes show that, although sometimes the targets may get lost in monitoring, they will still be found right after. The overall award is increased fast in the beginning, and is kept right under the highest level, for the majority of the simulation. Comparing with Figure 4.17, it can be seen that, having multiple agents brings significant improvement of the performance.

6.4.1.2 Comparative Study

The quantitative study is done to compare the performance of the proposed cooperative SSM and the non-cooperative SSM. Scenarios with n=2, 3, 5 and 7 targets had been studied, with $p_s = 60, 70, 80\%$, and with m=2, 3 and 5 pursuers. Each scenarios is

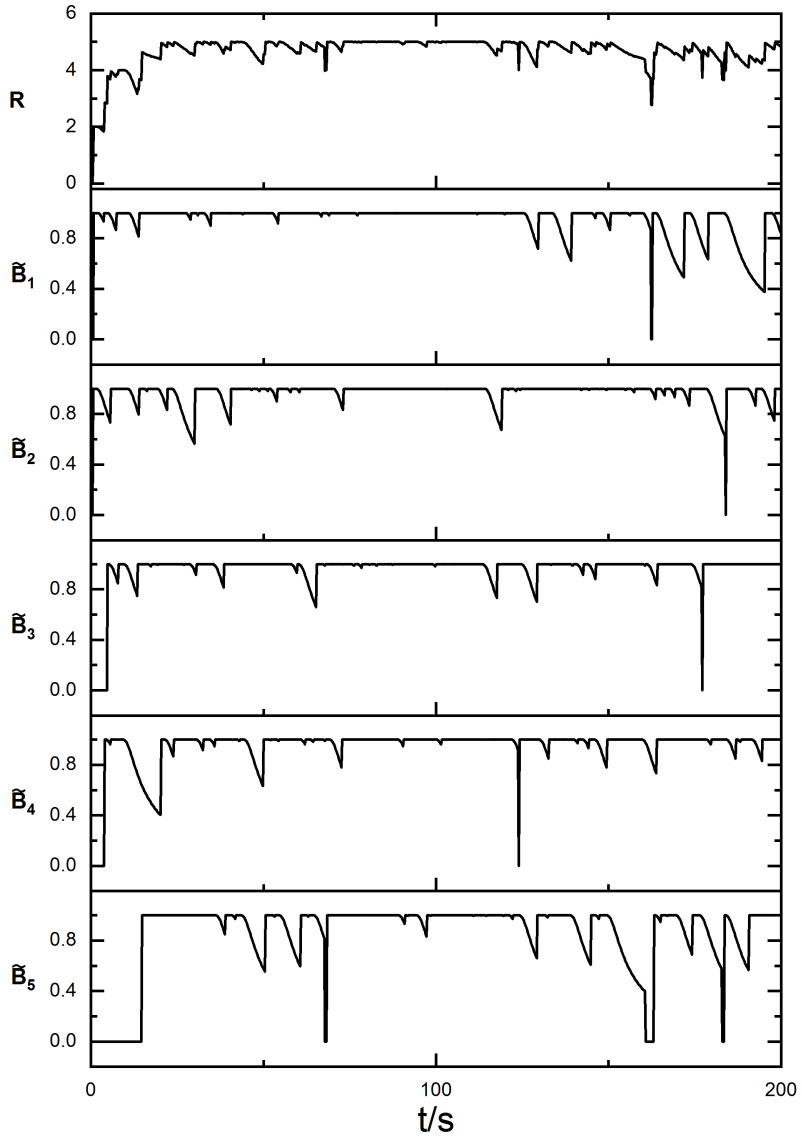


FIGURE 6.4: *belief probability* maintenance

simulated for the same number of cases and length of time as in the single pursuer simulation. Figure 6.5 shows the performances in each scenarios.

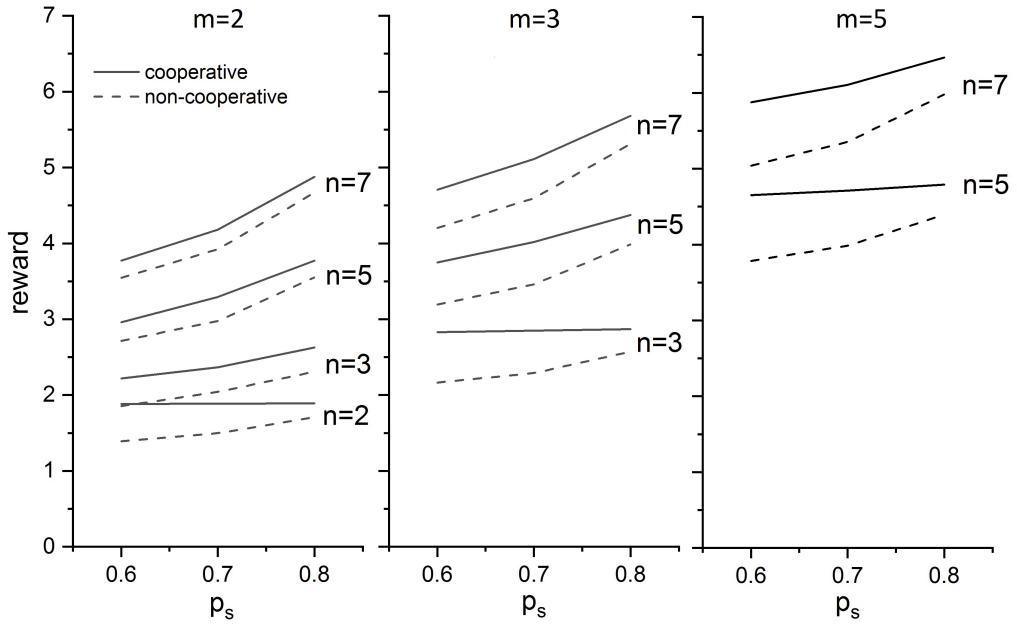


FIGURE 6.5: cooperative vs. non-cooperative

It shows that in every scenario, there is a dramatic improvement of performances if the cooperation is considered. It proves that, for the proposed distributed cooperative strategy planning, with only communication of measurement, it can achieve a better overall reward compared with not considering cooperation. This is consistent with Theorem 6.2.

Figure 6.6 illustrate the performances of the same scenarios, with imperfect sensors defined in Section 4.5.2. The performances decrease with possible sensing failures, but the cooperative SSM still has the advantage over the non-cooperative SSM. The same as mentioned in the single pursuer case, the false measurement can be treated with a filter, to reduce its negative influence.

The computational time of cooperative policy is in Table 6.1. The reader can refer to Table 4.1 for the computation time of non-cooperative policy.

number of agents \ number of targets	2	3	5	7
2 agents	0.41s	0.55s	0.70s	0.72s
3 agents	-	0.86s	1.21s	1.30s
5 agents	-	-	2.22s	2.80s

TABLE 6.1: computation time for cooperative policy planing (randomly moving targets)

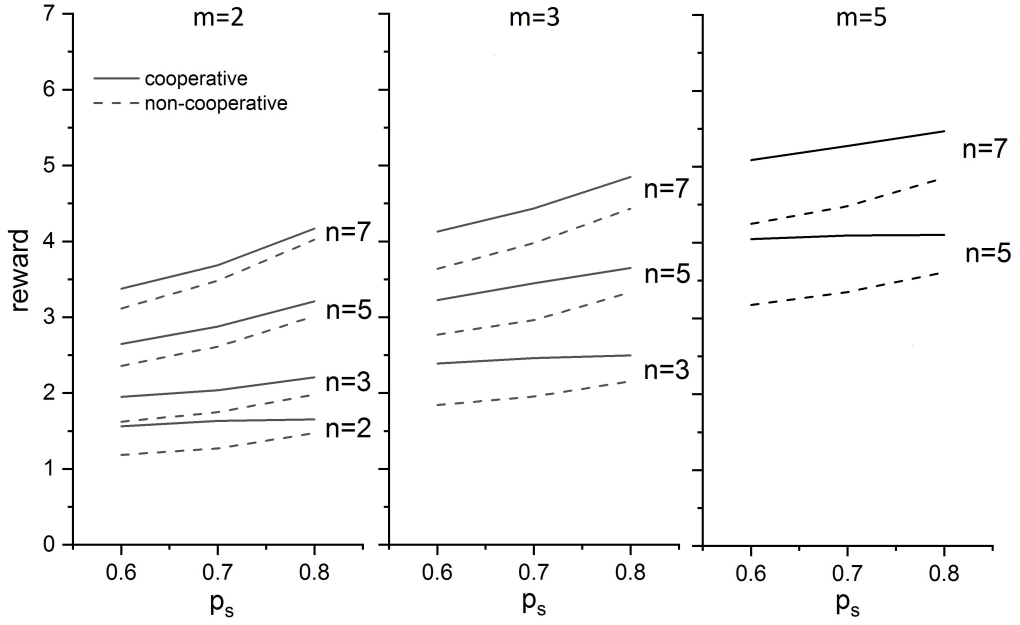


FIGURE 6.6: cooperative vs. non-cooperative with sensing failure

According to Table 6.1, each planning of cooperative policy takes 0.60 seconds in average, when there are 2 agents. It takes 1.12 and 2.51 seconds when there are 3 or 5 agents. It can be seen that, even though the planning is distributed, the computation time still grows with the number of the agents. This is because with more pursuers in the cooperation, the more agents needs to be considered in the computation of the equilibrium in Equation (6.10). Unfortunately, this shows that the proposed distributed strategy planning is not scalable. This problem will be addressed in Section 6.5.

6.4.2 Simulation of the Multiagent Simultaneous Search and Monitoring of Evasive Targets

6.4.2.1 Case Study

For the SSM of evasive targets, the same set-up of the environment and the properties of the players is taken, as in Section 5.6.1. There are 3 pursuers. The SSM is implemented cooperatively, as introduced in Section 6.3. The replanning is triggered at the same condition as in the single pursuer scenario, and it happens for all pursuers simultaneously.

Figure 6.7 to 6.10 are the snapshots of the simulation, where the solid lines are the plans or actual trajectories of the pursuers, the dash lines are the trajectories of the targets.

Figure 6.7 shows the plan of the pursuer at the initial time instant. For target 5 which is too far away from all of the pursuers, only pursuer 3 will go to attend it, thus leaving the

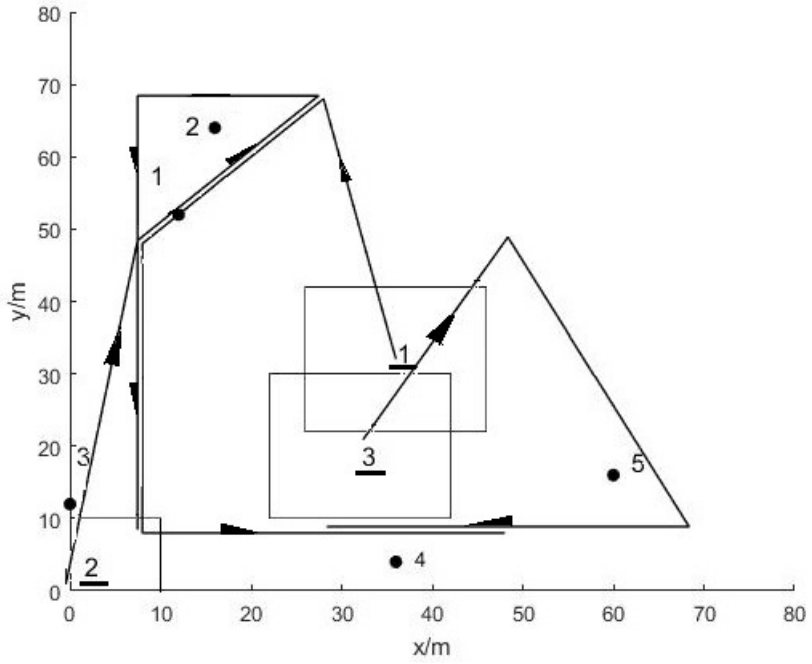


FIGURE 6.7: initial planning of the pursuer

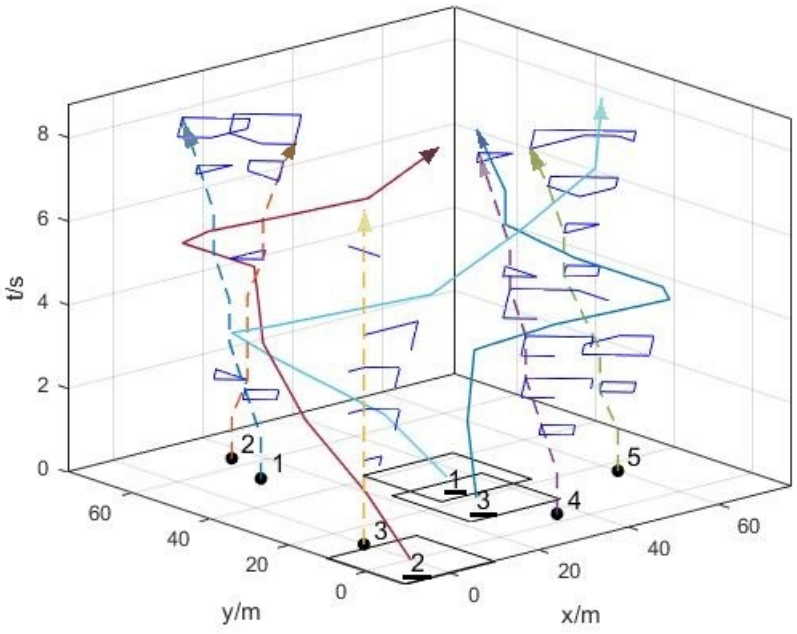


FIGURE 6.8: initial game playing

other pursuers to concentrate on the remaining targets. For the rest of the evaders, they will be visited by more than one agent at different time, thus limiting the *uncertainties* about them to grow. Figure 6.7 shows the evolution of the game for a few seconds from the beginning, corresponding to the initial plan in Figure 6.7. We can see that, all the targets can be detected at least once during this time period, to have the *uncertainties* about them to be cleared. For target 1, 2 and 3, they are detected for twice by different agents with a reasonable time interval, which successfully constrains the development of the *uncertainty* throughout the period.

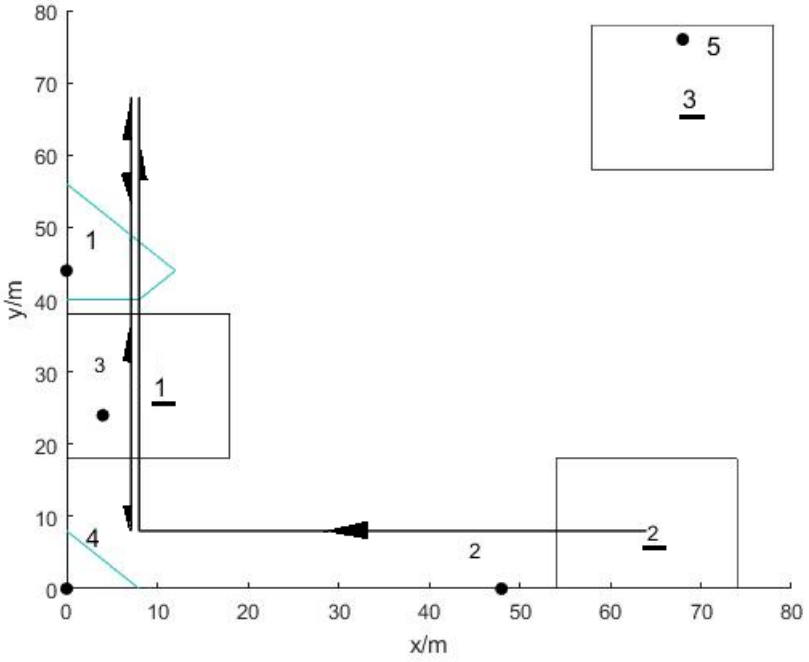


FIGURE 6.9: planning of the pursuer during the middle of the game

Figure 6.9 shows the pursuer planning in the middle of the game. When the target 5 is monitored by the pursuer 3, which are both far away from the other pursuers and targets, the target 3 stays unmoved to focus on target 5. The agent 2 goes to join the agent 1 to monitor target 1, 3 and 4. Figure 6.10 shows the development of the game after the planing in Figure 6.9 is made. It also shows that the target 1, 3, and 4 are detected by the pursuer 1 and 2 at different time, thus maintaining the corresponding *uncertainties* to a low level.

The snapshots show that, even without communication of planning, the agents can still coordinate their attempts of SSM. The cooperation is achieved through avoiding the redundant effort by letting each agent focus on different sets of targets, and through having the synergy in *uncertainty* reduction by letting different agents visit the same target in a timely order.

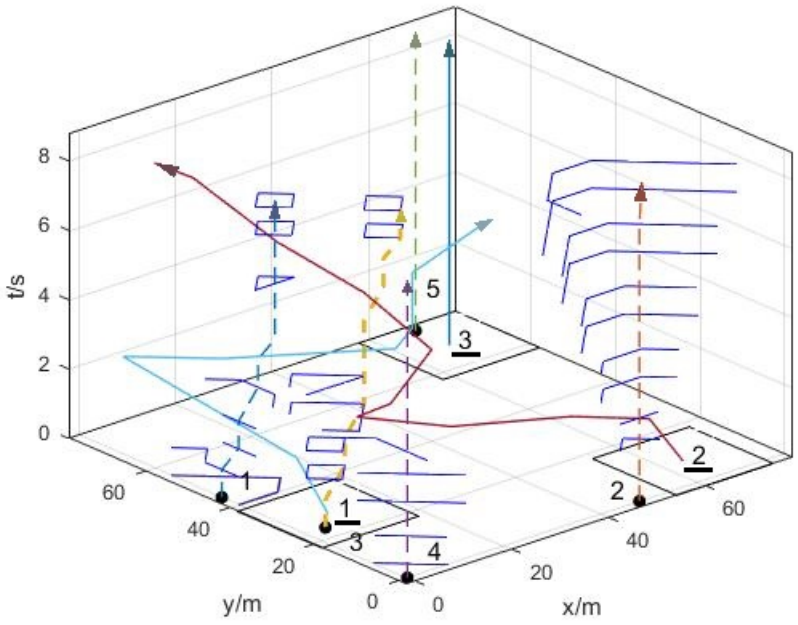


FIGURE 6.10: pursuit-evasion of targets

Figure 6.11 shows the development in a case study, of the *uncertainty* about each target and the total *uncertainty*. Compared with the single pursuer scenario (Figure 5.6), we can see that the individual *uncertainties* and the total *uncertainty* are controlled in a much lower level, and are cleared to zero much more frequently. Thus it shows that, by having multiple agents doing the SSM of evasive targets cooperatively, the performance can be dramatically improved.

6.4.2.2 Comparative Study

The quantitative study is done to compare the performance of the proposed cooperative SSM and the non-cooperative SSM. Scenarios with n=2, 3, 5 and 7 targets had been studied, with V_t = $1m/s$, $1.5m/s$, $2m/s$, $2.5m/s$, $3m/s$, $3.5m/s$, $4m/s$, and $4.5m/s$, and with m=2, 3 and 5 pursuers. Each scenario is simulated for the same number of cases and length of time as in the single pursuer simulation. Figure 6.12 shows the performances in each scenarios.

It shows that in every scenario, the performance of the cooperative SSM of the evasive targets is much better than the non-cooperative SSM, which proves the advantage of cooperative policy.

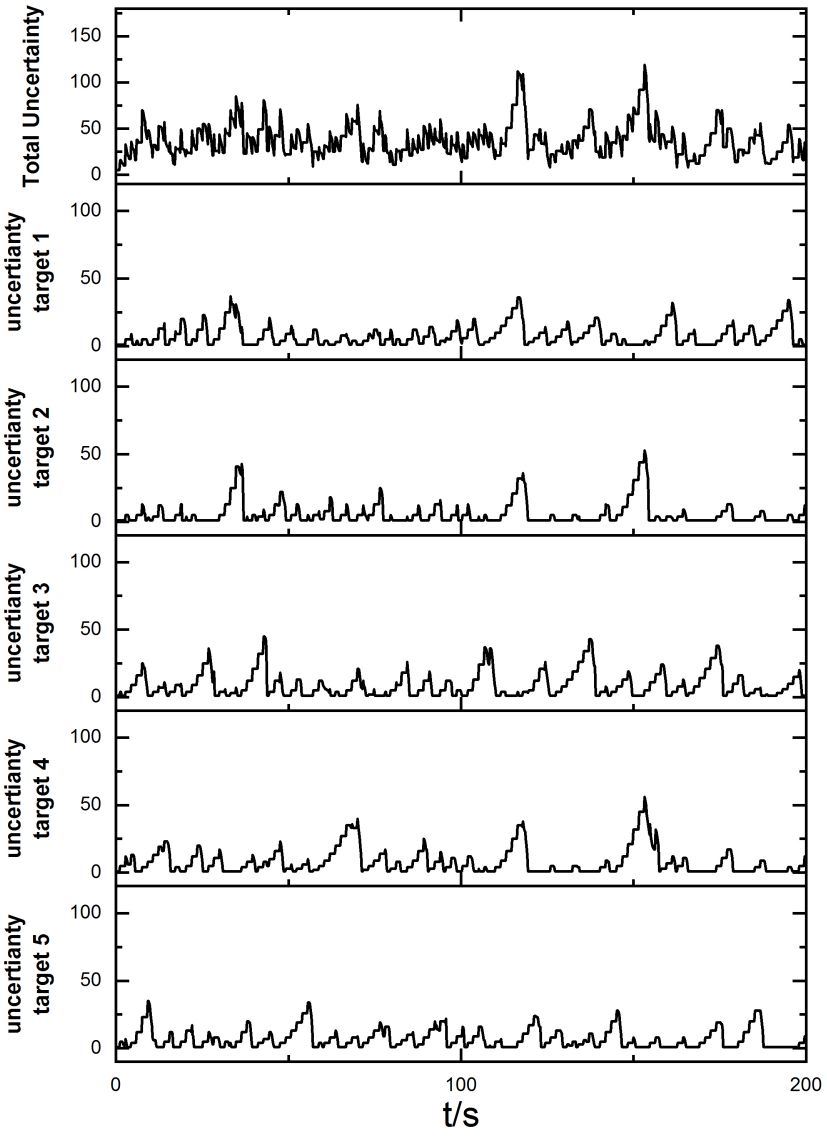


FIGURE 6.11: *uncertainty* reduction

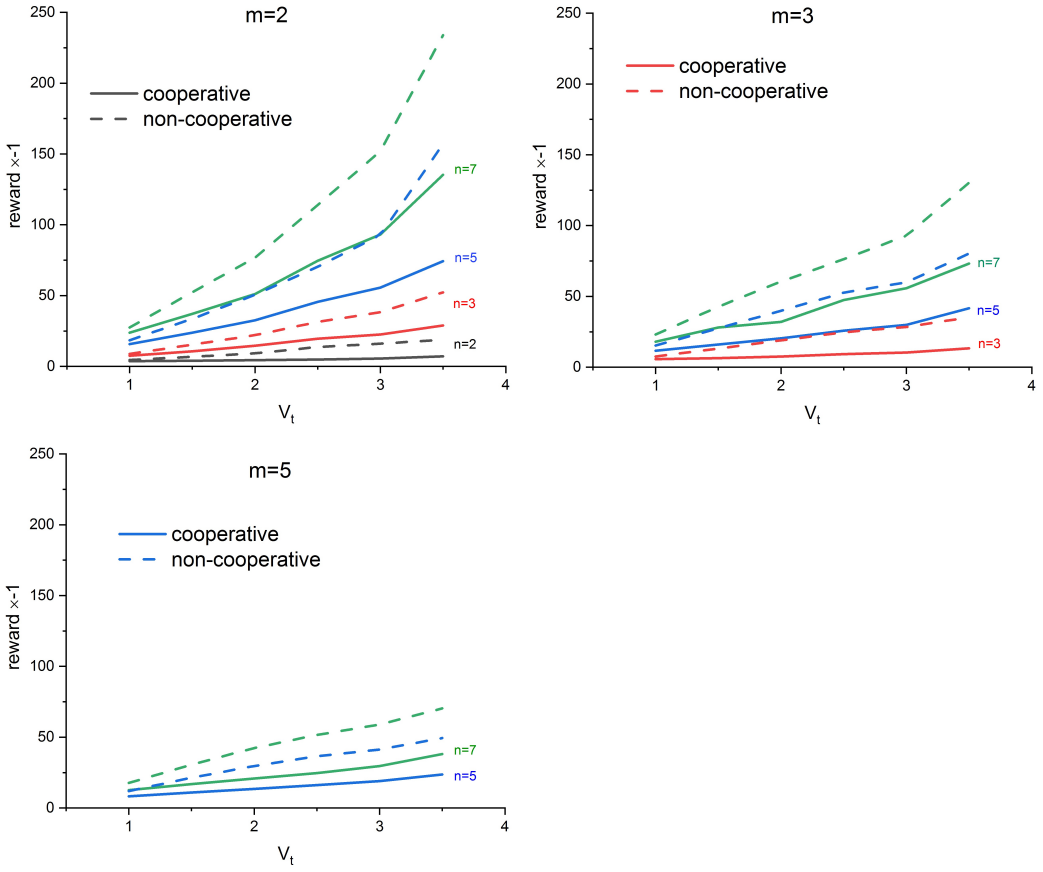


FIGURE 6.12: cooperative vs. non-cooperative

The computational time of cooperative policy is in Table 6.2. The reader can refer to Table 5.1 for the computation time of non-cooperative policy.

number of agents \ number of targets	2	3	5	7
2 agents	0.46s	0.62s	1.03s	1.44s
3 agents	-	0.97s	1.61s	2.01s
5 agents	-	-	2.87s	3.66s

TABLE 6.2: computation time for cooperative policy planing (evasive targets)

According to the simulation, each planning of cooperative policy takes 0.89 seconds in average, when there are 2 agents. It takes 1.53 and 3.27 seconds when there are 3 or 5 agents. It indicates the unscalability of the distributed policy planning. This is because of the same reason as in the cooperative SSM of randomly moving targets, that with the more agents in the game playing, when doing the distributed cooperative policy playing, each agent has to consider more fellow agents in the joint policy. This issue will be solved in the next Section.

6.5 Simultaneous Search and Monitoring with Limited Communication Range

It has been learned in the previous Section that, when the pursuers are planning the SSM cooperatively, the computation is not scalable with respect to the number of agents involved. To solve this problem, some facts are considered, that 1. the flying speed of an agent is constrained, which makes it of more interest for an agent to cooperate with pursuers close by; 2. in realistic applications, there should be a range limit for the communication and measurement between agents. Thus a communication range limit is introduced into the cooperation, so that only within such range, a pursuer can be aware of the presence and sense the location of a neighbouring agent, or receive the measurement information from it. This set-up not only includes the practical limits on sensing and communication, but also allows the agent to consider the fellow pursuers which are close enough to be of interest, thus bounding the number of agents to consider in the planning and reduce computation time.

Let L_c be the range limit of the communication. For agent ρ, let $\Gamma_\rho = \{\rho' : \rho' \in \Gamma, \rho' \neq \rho, |x_\rho(t) - x_{\rho'}(t)| \leq L_c\}$ be the set of agents which are within the communication range from agent ρ. In Chapter 3, the full communication between agents at all times is assumed. Under such assumption, the measurements are shared from the beginning, and with the fact that the initial estimation of the targets are shared, thus every agent can hold a common subjective state s_t or $\hat{s}_t$. Nevertheless, when the communication is limited, the sensing can not always be transmitted, and there should be difference in the perception of the environment by each agent.

Let s_t^ρ or $\hat{s}_t^\rho$ be the local subjective state hold by agent ρ, in the randomly moving or the evasive target scenario. From the definition of the subjective states in Section 6.2.1, Section 6.3, and the modelling in Chapter 3, we can see that, the subjective state is dependent on the sensing history. However, as introduced in Section 3.2.3, this work tries to reduce the load of communication between agents. Therefore, it does not require the pursuers to exchange their whole sensing histories when they communicate, and instead, they only send each other the current measurement information when within communication range. For the part of sensing histories which are not exchanged, they will cause each agent to hold different subjective state.

Then, each agent do the cooperative policy planning, considering only the local subjective state and the set of agent Γ_ρ:

$$\begin{aligned} \Pi^* &= \underset{\Pi}{argmax} V(\Pi, s^{\rho}_{t_i}, t_f) \\ \Pi &= \{\pi_{\rho'} : \rho' \in \Gamma_\rho\} \end{aligned} \tag{6.14}$$

Assume that there is a pseudo agent which can access the measurement from other agent at all time, and can thus hold a subjective state which considers all the sensing information. The actual achieved reward is evaluated based on the subjective state of this pseudo agent.

With the range limit on communication, the performance of cooperative SSM of randomly moving targets is shown in Figure 6.13 and Figure 6.14, with the communication range L_c = 50m or 30m. The performance of cooperative SSM of evasive targets is shown in Figure 6.15 and Figure 6.16.

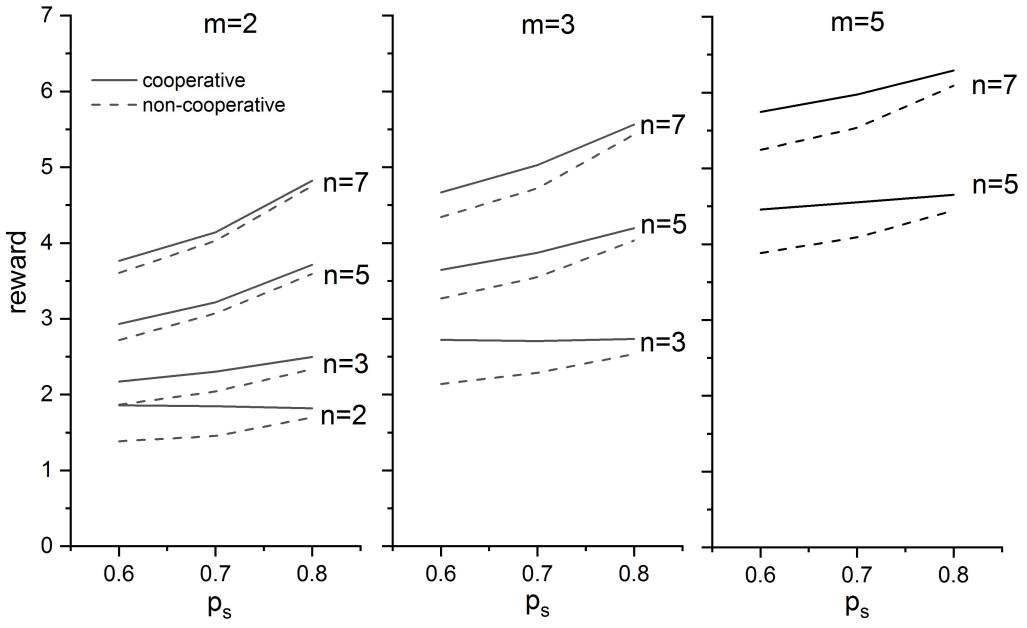


FIGURE 6.13: cooperative vs. non-cooperative when $L_c = 50m$ (randomly moving targets)

We can see from Figure 6.13 to Figure 6.16 that, compared with the multiagent SSM with full communication, the there is a slight degrade of performance when there is a limit on communication range. The smaller the range limit is, the lower the performance decreases. However, we can still see that, in almost every scenario, the performance of cooperative SSM is always higher than the non-cooperative SSM.

Table 6.3, 6.4, 6.5, and 6.6 shows the computation time of the cooperative policy planning in different scenarios, with different kinds of targets.

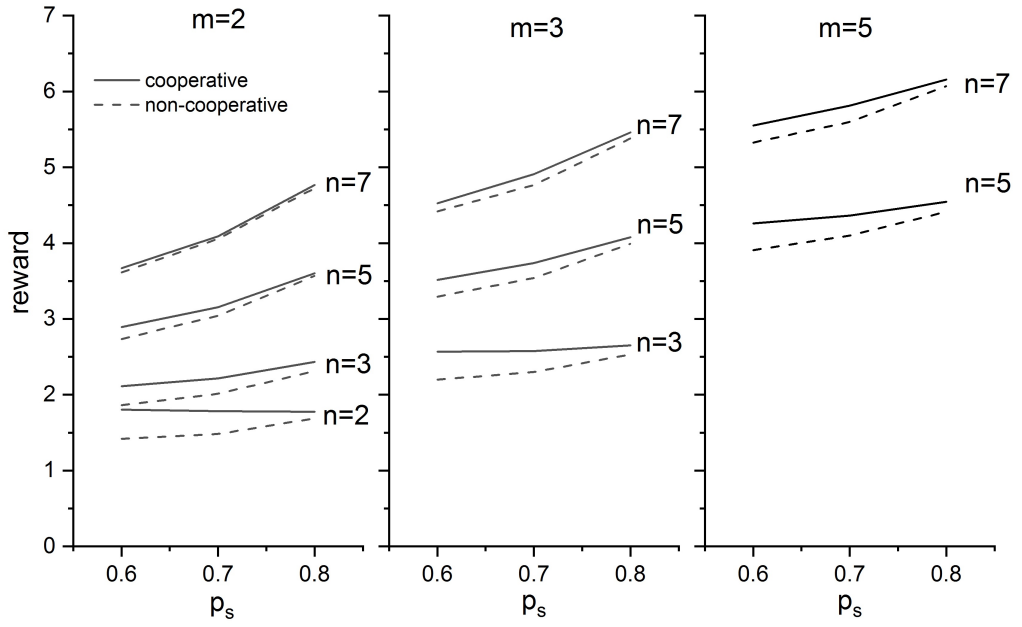


FIGURE 6.14: cooperative vs. non-cooperative when $L_c = 30m$ (randomly moving targets)

number of agents \ number of targets	2	3	5	7
2 agents	0.27s	0.44s	0.43s	0.55s
3 agents	-	0.48s	0.62s	0.77s
5 agents	-	-	1.11s	1.50s

TABLE 6.3: computation time for cooperative policy planing when $L_c = 50m$ (randomly moving targets)

number of agents \ number of targets	2	3	5	7
2 agents	0.23s	0.30s	0.38s	0.38s
3 agents	-	0.46s	0.55s	0.61s
5 agents	-	-	0.87s	1.00s

TABLE 6.4: computation time for cooperative policy planing when $L_c = 30m$ (randomly moving targets)

We can see from these tables that, with a limit on the communication range, there is a significant reduction of computation time needed for each cooperative planning. With a tighter limit, the reduction is higher. This proves the main benefit of introducing a communication range, which bounds the number of fellow agents to consider in each planning, and allow each agent to cooperative with only nearby agents which are of more benefit than the faraway ones.

Thus it has been known that, in a more realistic scenario where there is a range limit on

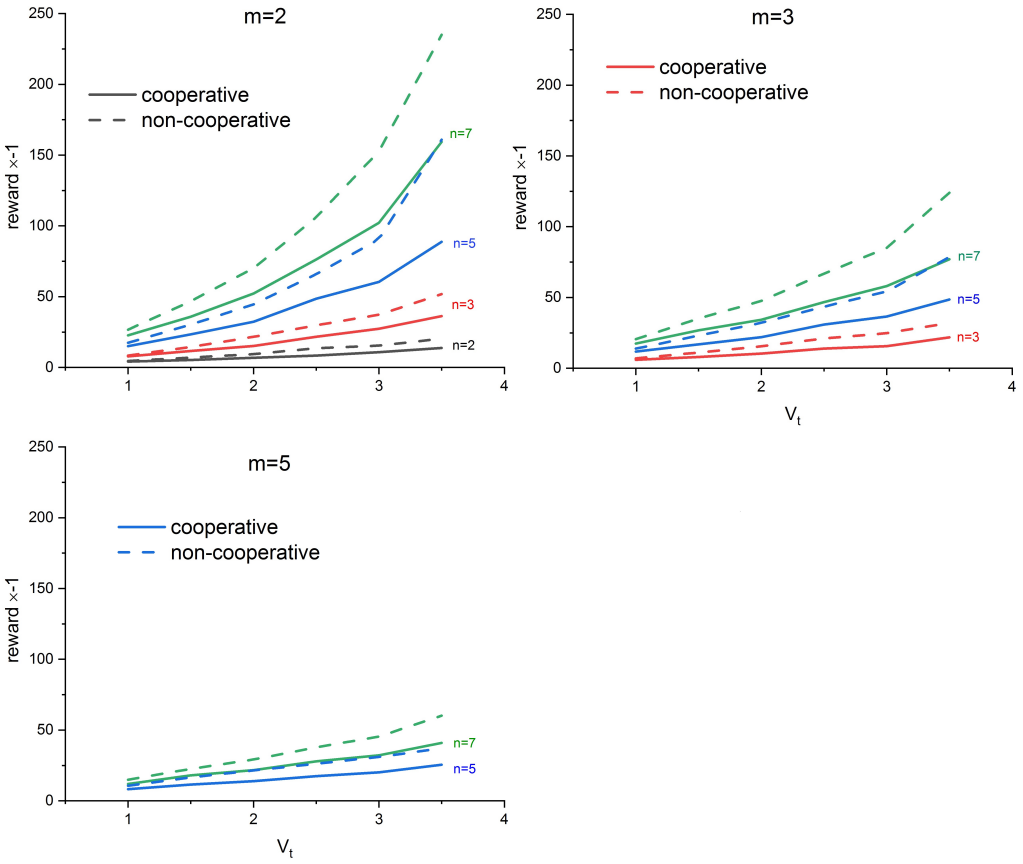


FIGURE 6.15: cooperative vs. non-cooperative when $L_c = 50m$ (evasive targets)

number of agents \ number of targets	2	3	5	7
2 agents	0.46s	0.63s	1.01s	1.29s
3 agents	-	0.79s	1.29s	1.65s
5 agents	-	-	1.95s	3.14s

TABLE 6.5: computation time for cooperative policy planing when $L_c = 50m$ (evasive targets)

the communication and measurement between each pursuer, without doing any significant modification to the planning method or increasing the workload of communication, the cooperative strategy planning can still make the team of agents to achieve higher performance than the non-cooperative planning, and the computation efficiency is largely improved. Also, this range limit can be imposed to be tighter than the actual limit for the purpose of improving the computational efficiency and making the planning more scalable.

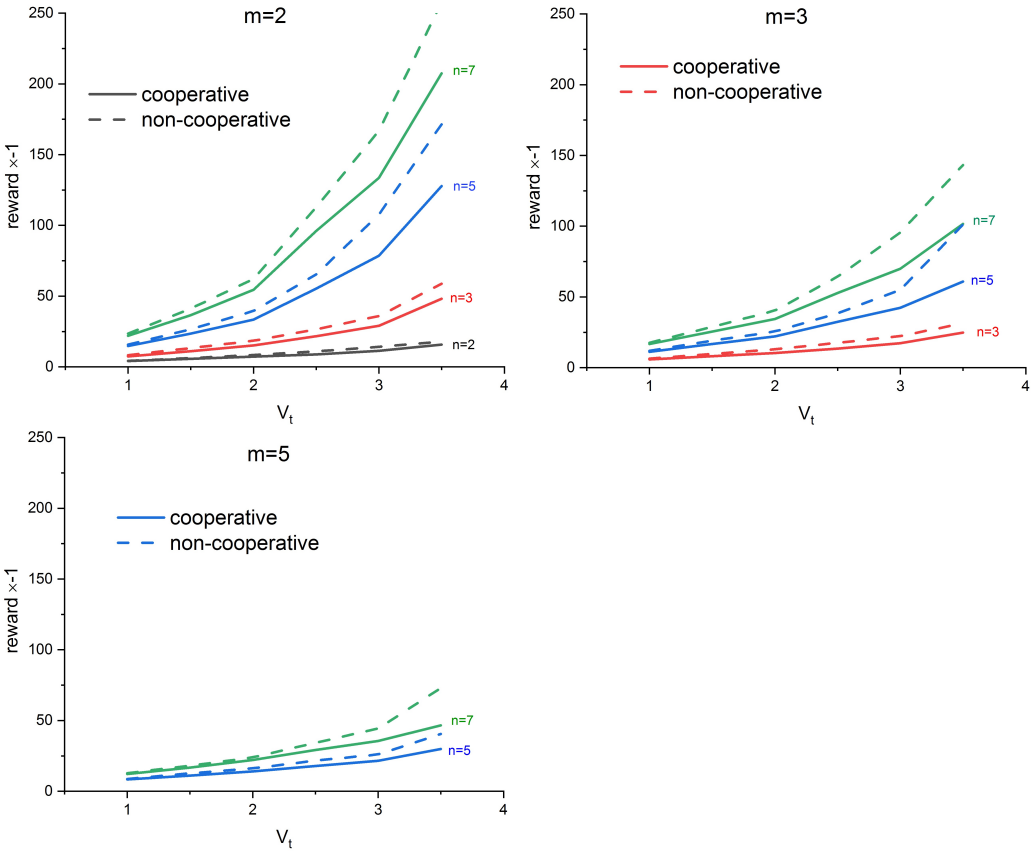


FIGURE 6.16: cooperative vs. non-cooperative when $L_c = 30m$ (evasive targets)

number of agents \ number of targets	2	3	5	7
2 agents	0.41s	0.50s	0.72s	1.05s
3 agents	-	0.53s	0.87s	1.32s
5 agents	-	-	1.41s	1.95s

TABLE 6.6: computation time for cooperative policy planing when $L_c = 30m$ (evasive targets)

6.6 Considering the Manoeuvrability of the Pursuer

To consider the influence of the UAV model on this work, the minimum turning radius is considered for the quantitative study. Let $r_c = 5m$. All the above quantitative simulations are done again, to prove the validity of the above conclusions in a practical application. The results are from Figure 6.17 to Figure 6.22.

Compared with Section 6.4 and 6.5, we can see that, with a more practical agent model, the advantage of the cooperative SSM still stands, compared with the non-cooperative SSM. Thus all the discussions and conclusions in the above sections still uphold.

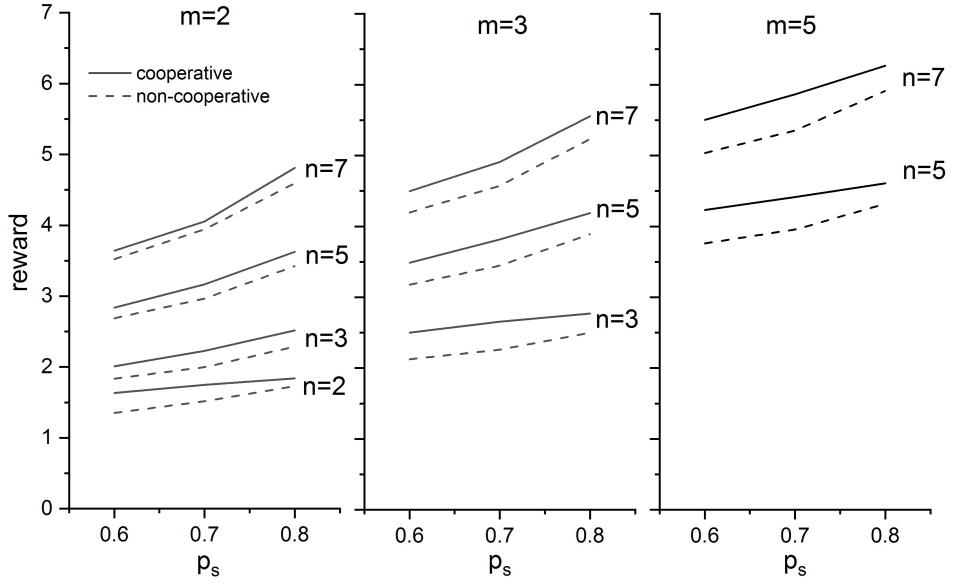


FIGURE 6.17: cooperative vs. non-cooperative with full communication and $r_c = 5m$ (randomly moving targets)

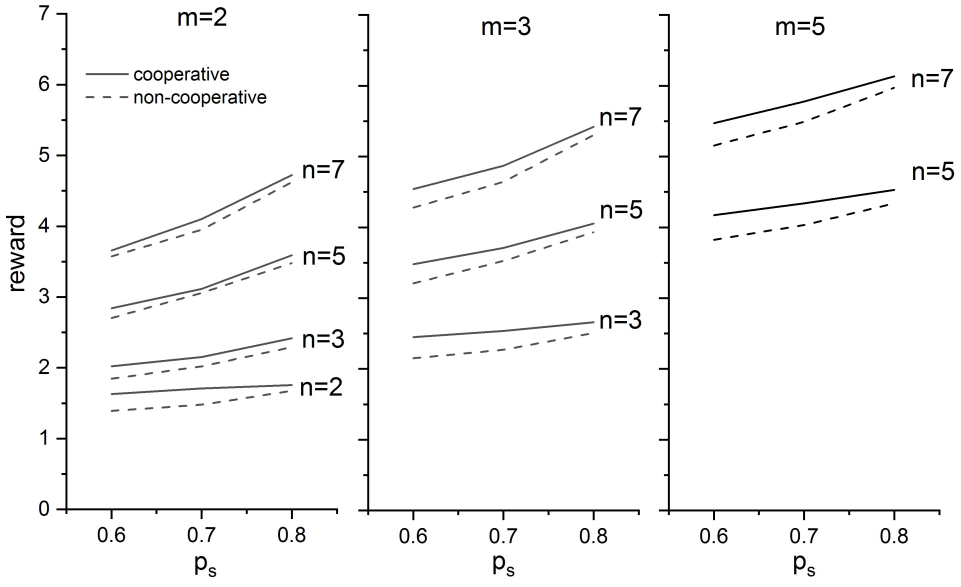


FIGURE 6.18: cooperative vs. non-cooperative when $L_c = 50m$ and $r_c = 5m$ (randomly moving targets)

6.7 Exploring the limitation of SSM

After proving the efficiency of multi-agent SSM in previous sections. The same as in section 4.5.4 and 5.6.4, in this section, the simulations are expanded to some more extreme situations, to find out the practical limitations on SSM. For the simulation in Section 6.4.1.2, 6.4.2.2, and 6.5, we expand the p_s or V_t of targets and the size of

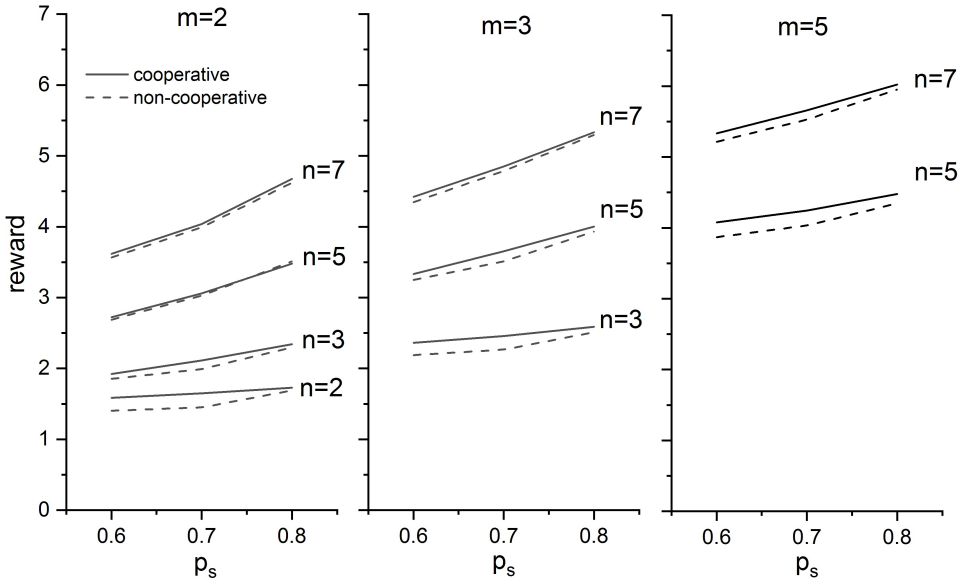


FIGURE 6.19: cooperative vs. non-cooperative when $L_c = 30m$ and $r_c = 5m$ (randomly moving targets)

environment, as the same way in Section 4.5.4 and 5.6.4. The results are shown from Figure 6.23 to 6.40:

It can be seen from Figure 6.23 to 6.40 that, with more active or faster targets, or in a bigger environment, the performance of SSM degrades. Similar to the single pursuer scenario, when $p_s \leq 40\%$ or $V_t > 5m/s$, or when the environment is $180m \times 180m$, the multi-agent SSM has reached its limitations.

For the cooperative SSM of randomly moving targets, when the limitations are reached, the average performance of each agent is close to that of monitoring one single target. Similar to Section 4.5.4, this also shows that, in the adverse conditions, the strategy with the highest reward is for each agent to keep monitoring the first target it finds. We can also see from Figure 6.26 to 6.31 that, the cooperative SSM is less sensitive to the extreme conditions, especially when there is full or longer range of communication. This is because, with cooperation, the agents can be efficiently scattered to cover more area and targets. This also shows the advantage of cooperative SSM against non-cooperative version.

For the cooperative SSM of evasive targets, when close to or beyond the limitations, the total uncertainty grows uncontrolled, which is of the same reason stated in Section 5.6.4. We can also see from Figure 6.32 to 6.40 that, with full communication, the cooperative SSM is less prone to be affected by the increase of the size of environment. This also proves the effectiveness of cooperation in the SSM of evasive targets.

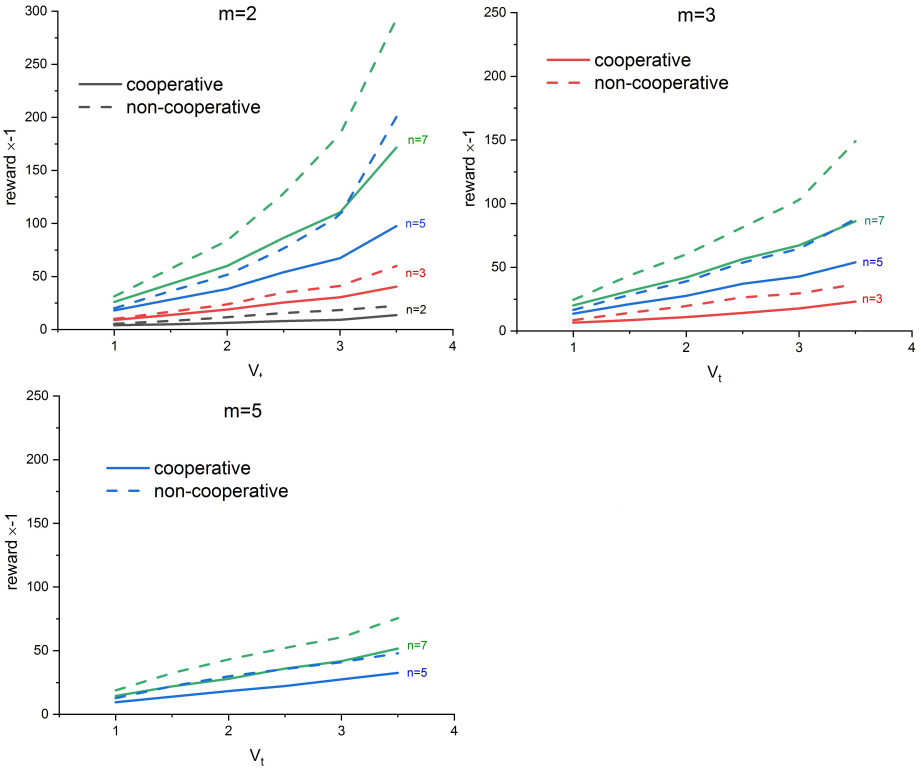


FIGURE 6.20: cooperative vs. non-cooperative with full communication and $r_c = 5m$ (evasive targets)

6.8 Conclusion

The cooperative SSM is solved in a distributed and online way. The cooperative equilibrium is applied as the main concept as solution. For the scenario with randomly moving targets, solving the strategy planning of the SSM is a Dec-POMDP. The idea of Partial Open-Loop Feeback Control and heuristic reactive policy reconstruction are combined in a novel way. Thus a intuitive cooperative policy planning can be designed, which allows some intuitive heuristic methods to be incorporated and can be fast to compute. The cooperation can still be considered implicitly. For the scenario with evasive targets, same simplification in the single pursuer case can still be applied, which also simplified the problem to be a cooperative dynamic guaranteed search.

In the simulation of the SSM of randomly moving targets, the agents can divide the unknown areas and known targets, and attend each part separately, thus avoiding overlapping efforts. In the simulation of the SSM of evasive targets, the agents not only divide the targets to avoid redundancy, but also have synergy by visiting a certain target at a suitable time interval. For both scenarios, in the quantitative study, the cooperative SSM shows significant improvement of the performance, compared with the non-cooperative SSM, which validates the advantage of cooperation.

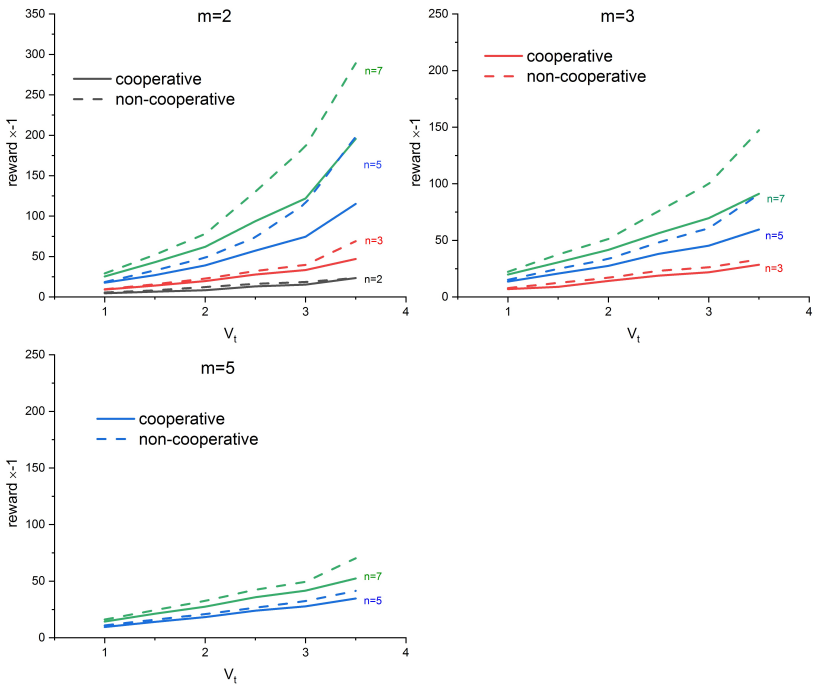


FIGURE 6.21: cooperative vs. non-cooperative when $L_c = 50m$ and $r_c = 5m$ (evasive targets)

To address the problem of scalability of the distributed strategy planning, but also to include practical constraints, the limit on the range of communication is introduced. The simulation results show that, even with a communication range, the performance of the cooperative SSM is still better than the non-cooperative SSM. The computation time is reduced in such way to achieve scalability. The simulation with a realistic agent model also supports the above conclusions.

The practical limitation on multi-agent SSM is studied in Section 6.7. The limitations in the single pursuer SSM cases still applies in multi-agent scenarios. However, in the extreme conditions, the cooperative SSM shows its better robustness compared with non-cooperative SSM, which validates the effectiveness of the designed cooperation strategy. And as shown in Section 6.4.1.2 and 6.4.2.2, the cooperative SSM can be efficient under reasonable conditions.

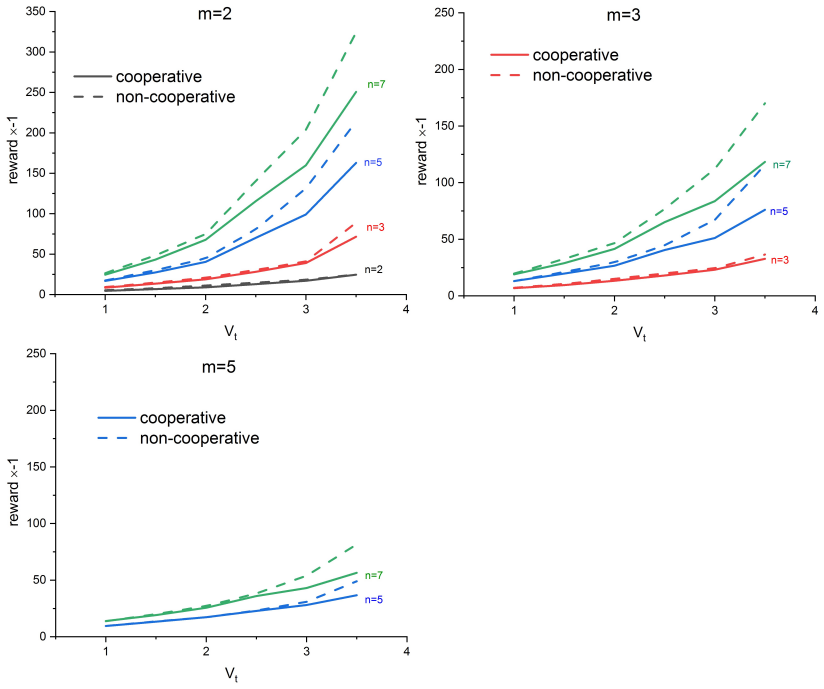


FIGURE 6.22: cooperative vs. non-cooperative when $L_c = 30m$ and $r_c = 5m$ (evasive targets)

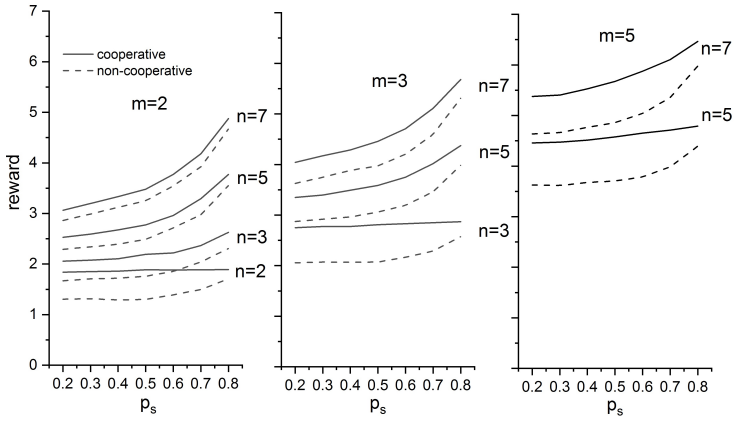


FIGURE 6.23: cooperative vs. non-cooperative with unlimited communication, and with environment width $L = 100m$ (randomly moving targets)

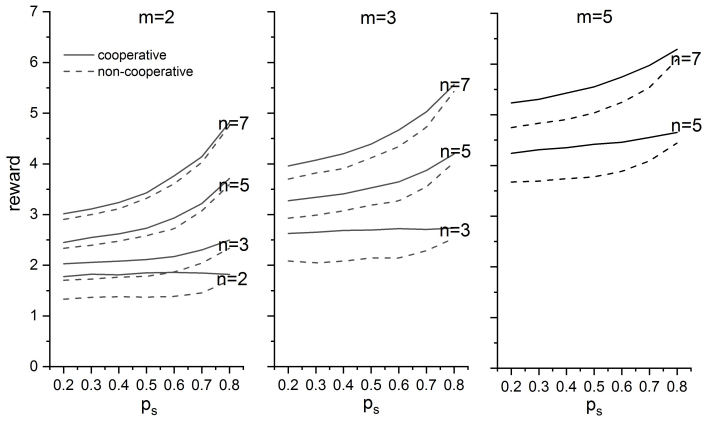


FIGURE 6.24: cooperative vs. non-cooperative with $L_c = 50m$, and with environment width $L = 100m$ (randomly moving targets)

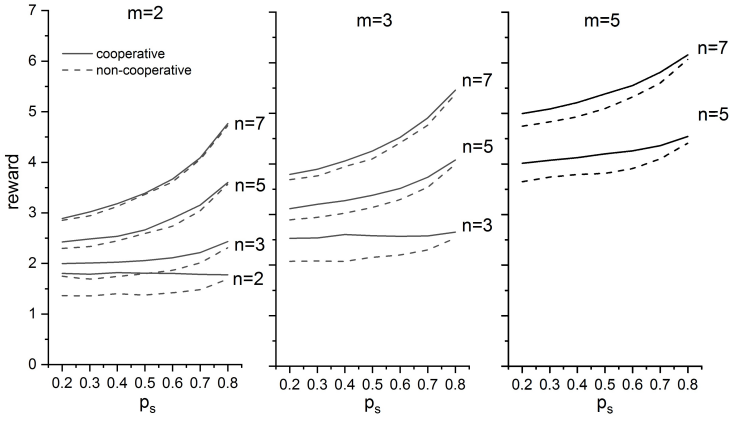


FIGURE 6.25: cooperative vs. non-cooperative with $L_c = 30m$, and with environment width $L = 100m$ (randomly moving targets)

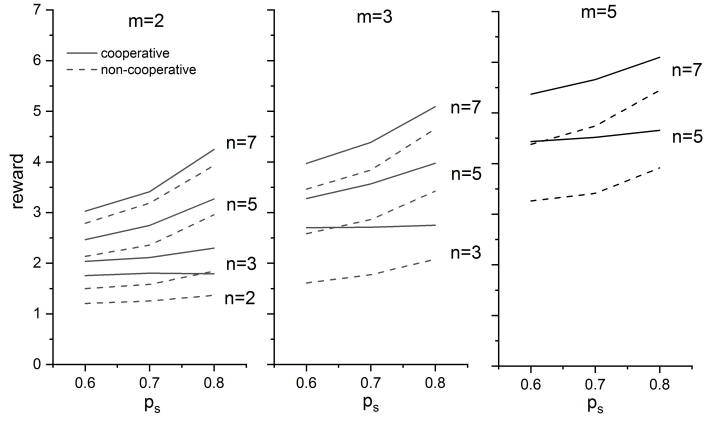


FIGURE 6.26: cooperative vs. non-cooperative with unlimited communication, and with environment width $L = 140m$ (randomly moving targets)

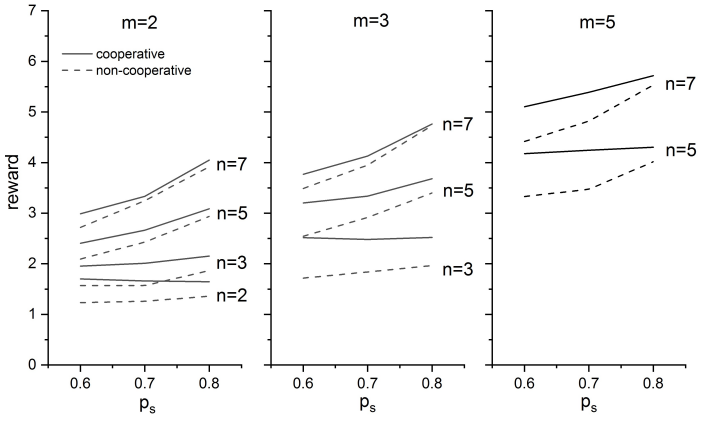


FIGURE 6.27: cooperative vs. non-cooperative with $L_c = 50m$, and with environment width $L = 140m$ (randomly moving targets)

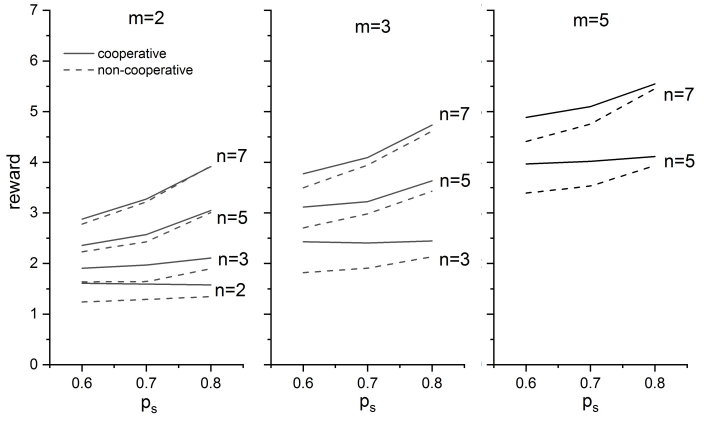


FIGURE 6.28: cooperative vs. non-cooperative with $L_c = 30m$, and with environment width $L = 140m$ (randomly moving targets)

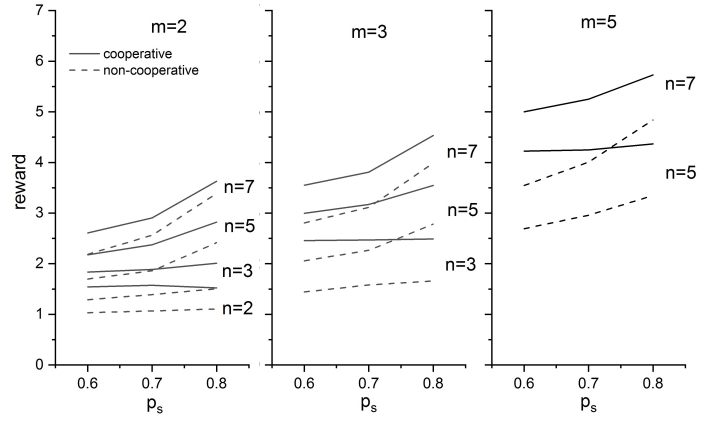


FIGURE 6.29: cooperative vs. non-cooperative with unlimited communication, and with environment width $L = 180m$ (randomly moving targets)

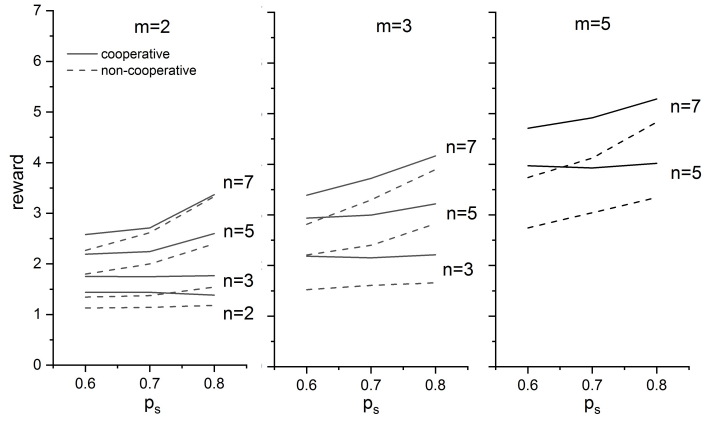


FIGURE 6.30: cooperative vs. non-cooperative with $L_c = 50m$, and with environment width $L = 180m$ (randomly moving targets)

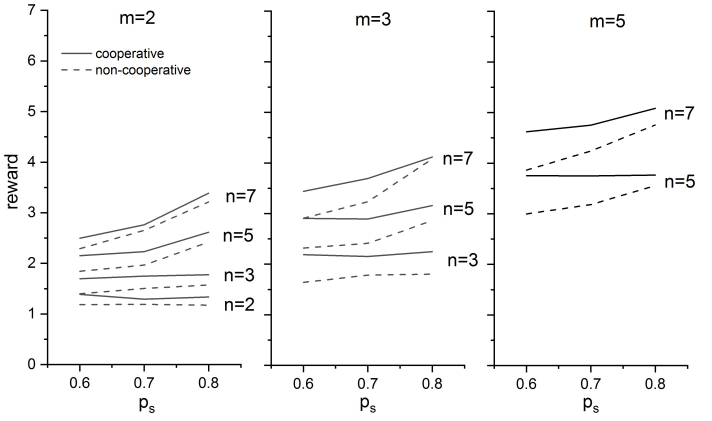


FIGURE 6.31: cooperative vs. non-cooperative with $L_c = 30m$, and with environment width $L = 180m$ (randomly moving targets)

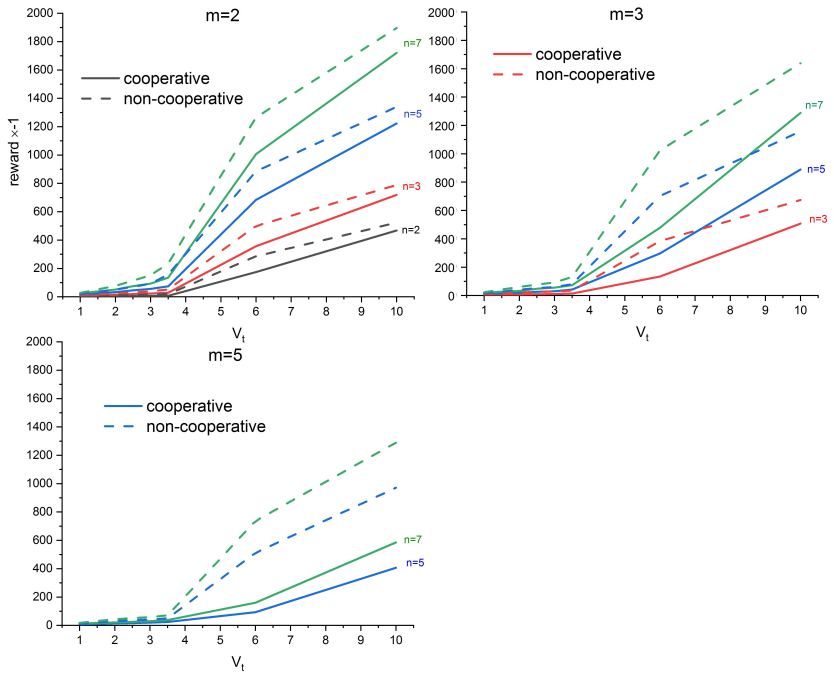


FIGURE 6.32: cooperative vs. non-cooperative when full communication, and with environment width $L = 80m$ (evasive targets)

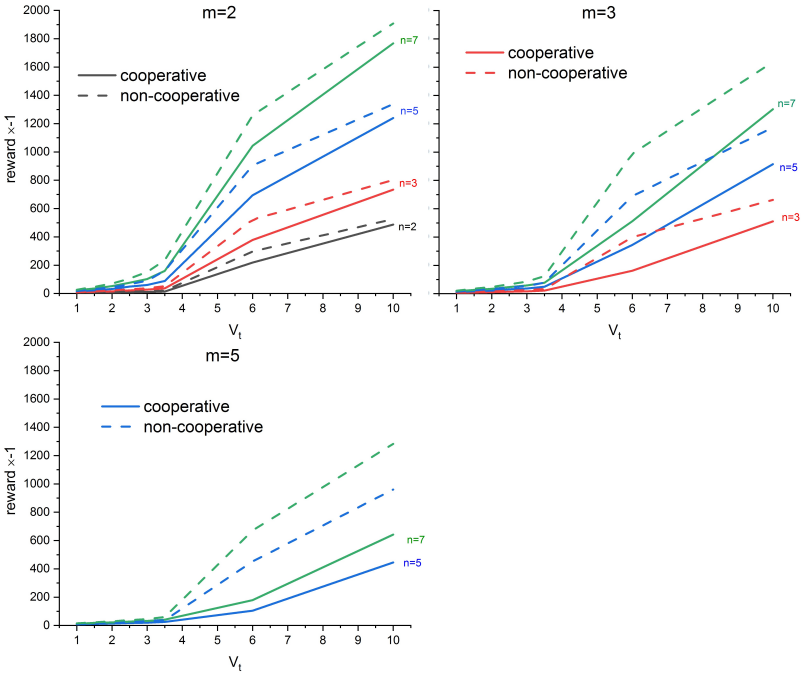


FIGURE 6.33: cooperative vs. non-cooperative with $L_c = 50m$, and with environment width $L = 80m$ (evasive targets)

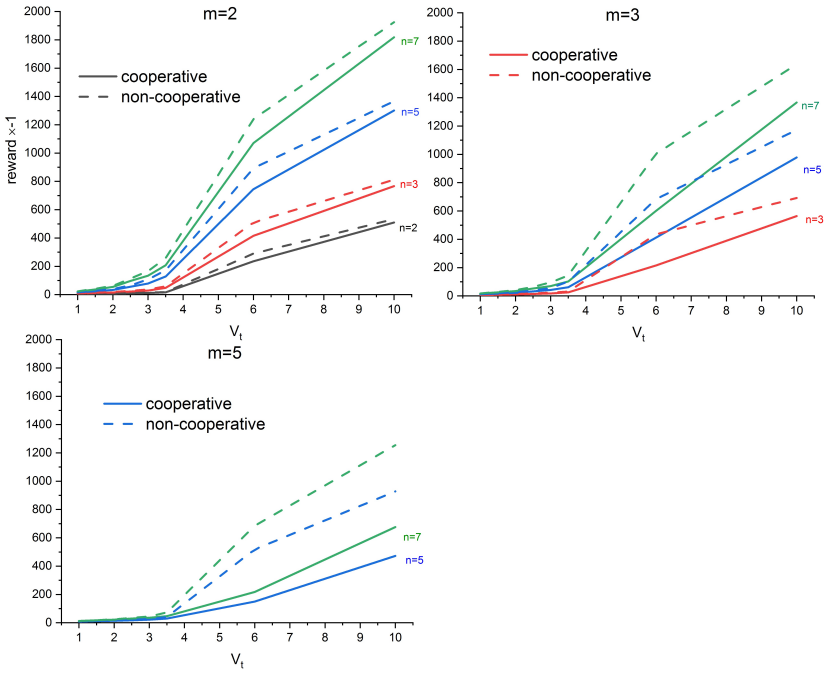


FIGURE 6.34: cooperative vs. non-cooperative with $L_c = 30m$, and with environment width $L = 80m$ (evasive targets)

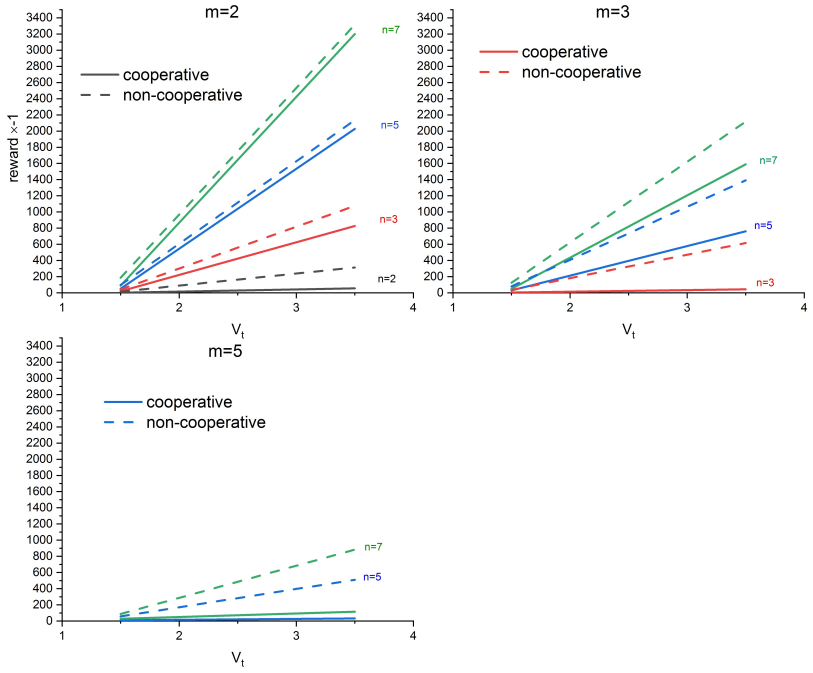


FIGURE 6.35: cooperative vs. non-cooperative when full communication, and with environment width $L = 140m$ (evasive targets)

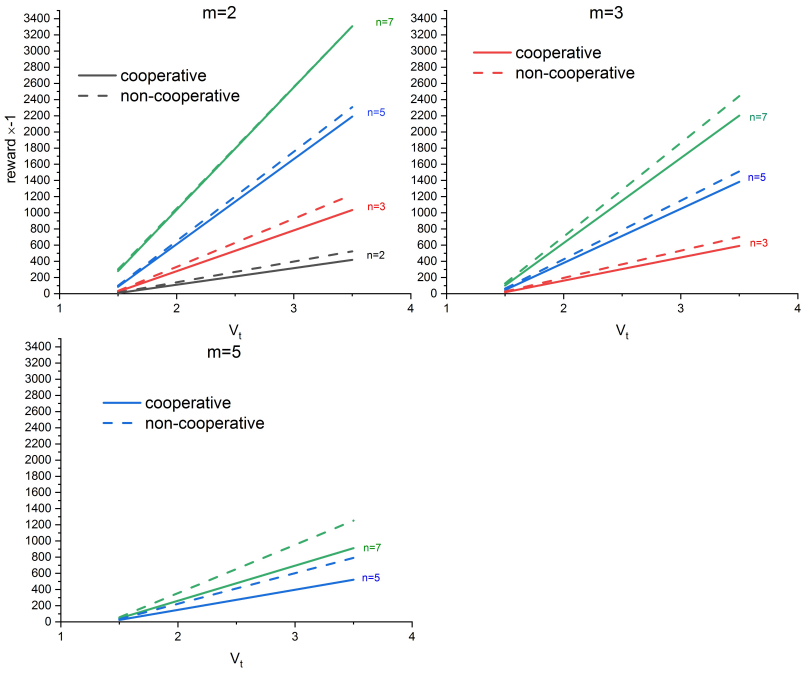


FIGURE 6.36: cooperative vs. non-cooperative with $L_c = 50m$, and with environment width $L = 140m$ (evasive targets)

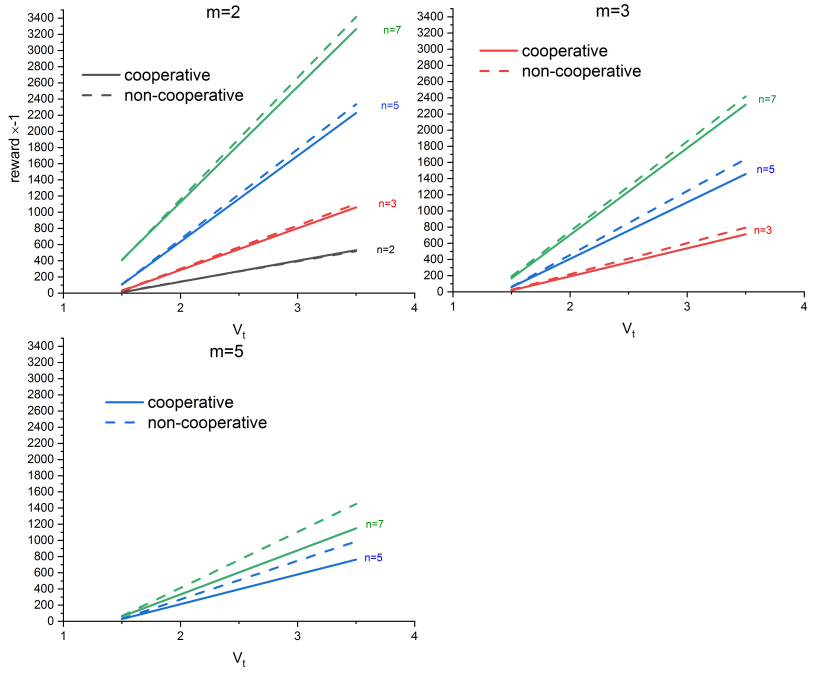


FIGURE 6.37: cooperative vs. non-cooperative with $L_c = 30m$, and with environment width $L = 140m$ (evasive targets)

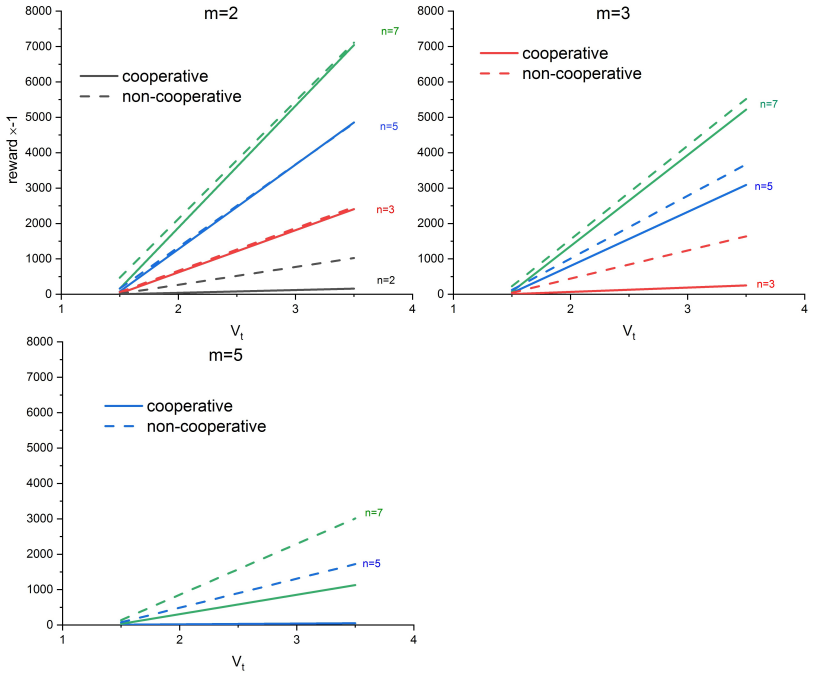


FIGURE 6.38: cooperative vs. non-cooperative when full communication, and with environment width $L = 180m$ (evasive targets)

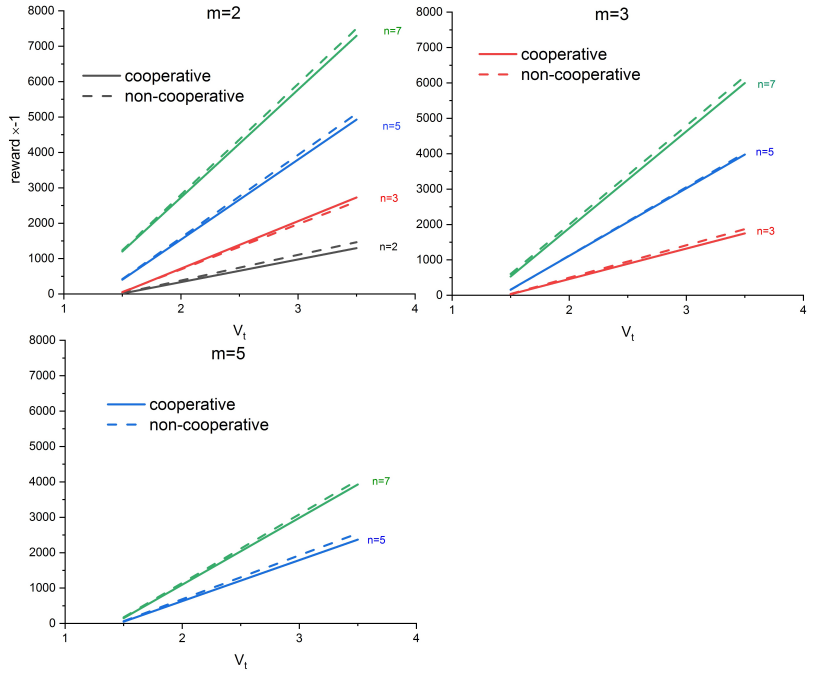


FIGURE 6.39: cooperative vs. non-cooperative with $L_c = 50m$, and with environment width $L = 180m$ (evasive targets)

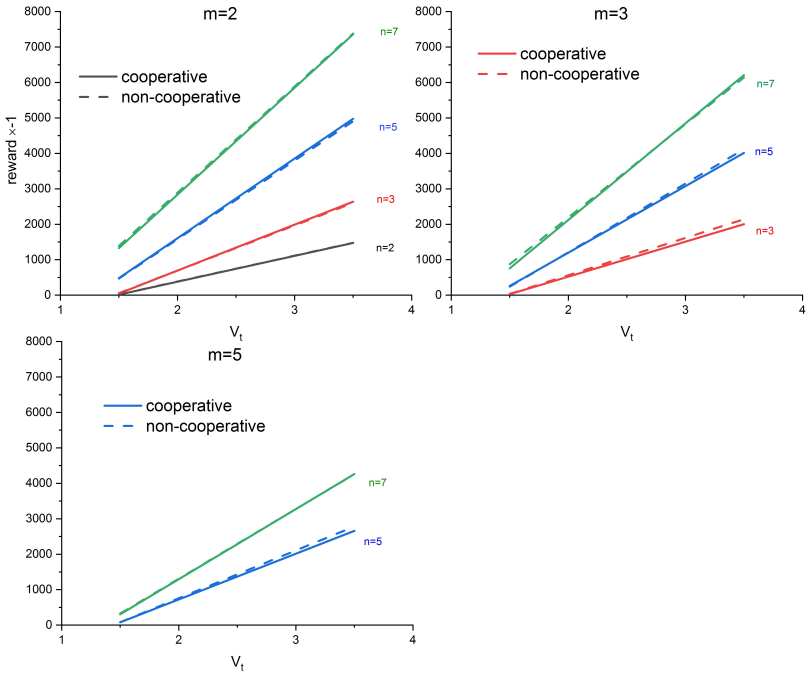


FIGURE 6.40: cooperative vs. non-cooperative with $L_c = 30m$, and with environment width $L = 180m$ (evasive targets)

Chapter 7

Conclusion and Future Work

7.1 Conclusion

From a broad background of robot search and pursuit evasion, the isolation between the problems of search and monitoring is discussed. For the initial information about the targets, different uncertainty level of them leads to the differences between these two problems. Nevertheless, an idea is conceived in this work, that in a situation when the targets are dynamic and the pursuers only have partial observation, search and monitoring are both needed at the same time, to obtain and maintain the updated overall information. Instead of building the combination as a trade-off, a novel concept of combining Search and Monitoring in a synergistic perspective is proposed. This allows the agents to better consider the dynamics of the problem, such as the targets changing between known and unknown. The combination of Search and Monitoring is done by building a united goal for the mission, which implicitly encourages the search and monitoring to be done simultaneously and cooperatively.

To solve this simultaneous search and monitoring problem, the main effort of this work is on how to have a scalable and practical solution, to have an online and distributed strategy planning. This work is split into two steps. The first step is to solve the single pursuer SSM of randomly moving or evasive targets. The second step is to extend the single pursuer SSM to the multiple pursuers scenario.

For the single pursuer SSM of randomly moving targets, which is built as a POMDP, the innovative policy reconstruction makes it easy to incorporate heuristics into the policy design, and generates a heuristic reactive policy. Compared with the policy of fixed sequence of actions which is not reactive, or the hybrid policy which is more rigorously designed, the heuristic reactive policy shows better performance and has

practical computational efficiency. For the single pursuer SSM of evasive targets, because of the intractability of precisely solving it as a POSG, an assumption are made about the information available to the targets. This simplifies the problem to a dynamic guaranteed search, which also showed advantage over a conventional method.

For the scenario with multiple pursuers, the coordination is achieved by solving a cooperative equilibrium in a distributed way. For the multiple pursuers SSM with randomly moving targets, which is a Dec-POMDP, the concept of partial open-loop feedback control and heuristic reactive policy are originally combined, which allows an intuitive strategy design without ignoring the cooperation. For the multiple pursuers SSM with evasive targets, which is a POSG, the same assumption in the single pursuer scenario simplifies the cooperative strategy planning to cooperative path planning. In both cases, the cooperative SSM performs better than the non-cooperative SSM. To improve the scalability of the distributed planning, and also to include practical limitation, the maximum communication range is imposed. With a range limit on communication, the cooperative SSM still has better performance than non-cooperative SSM, but the computation time is significantly reduced.

For all the research above, it is also tested with a more realistic agent model, of which the minimum turning radius is considered. And through the simulation, we can see that even when the agents have limited manoeuvrability, all the conclusions above still uphold.

Besides developing and evaluating efficient SSM strategies, for the single or multiple pursuer SSM of randomly moving or evasive targets, the practical limitations on its effectiveness are also studied. The detailed limitations are obtained by testing the SSM with more active targets and wider size of environment, until the advantage of SSM degrades to be trivial. These limitations are just reference for predicting and evaluating the performance of SSM in realistic situations, which does not undermine the benefit of SSM studied in this thesis.

In summary, the SSM problems in different scenarios have been tackled under moderation conditions and with feasible computation costs. Different measures are taken in each scenario, to solve the formerly intractable problem online and in real time. Under moderate conditions, the solutions proposed in this work have better performance compared with baseline methods.

7.2 Future Work

For the SSM of randomly moving targets, the underlying POMDP or Dec-POMDP are both intractable. For the practical application, further research on how to improve the computational efficiency is vital to its success. This can be done by tailoring the planning to each specific application, such as designing heuristic functions for value estimation. For the SSM of evasive targets, more realistic target model can be studied to accurately predict the target behaviours, thus to relax the current worst case assumption without inducing excessive computation load. For the multi-agent SSM, a practical communication model should be considered and included into the policy planning, for the application of multi-agent SSM on realistic systems. The concept of sensor network can be applied in multi-agent SSM, to allow information sharing without full communication. The scenarios when the initial number and property of targets are unknown should also also be studied, which can be combined with machine learning techniques.

The experimental validation of this work is on progress, which will be included in a Journal paper. The author will always look for chances to apply the SSM algorithm in real robot system to push the current boundaries.

Bibliography

[1] Sara Bernardini, Maria Fox, and Derek Long. Combining temporal planning with probabilistic reasoning for autonomous surveillance missions. *Autonomous Robots*, 41(1):181–203, 2017.

[2] Eric W Frew. Combining area patrol, perimeter surveillance, and target tracking using ordered upwind methods. In *2009 IEEE International Conference on Robotics and Automation*, pages 3123–3128. IEEE, 2009.

[3] Yan Jin, Yan Liao, Ali A Minai, and Marios M Polycarpou. Balancing search and target response in cooperative unmanned aerial vehicle (uav) teams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(3): 571–587, 2005.

[4] Eric W Frew and Jack Elston. Target assignment for integrated search and tracking by active robot networks. In *2008 IEEE International Conference on Robotics and Automation*, pages 2354–2359. IEEE, 2008.

[5] Jack Elston and Eric W Frew. Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks. In *2008 IEEE International Conference on Robotics and Automation*, pages 170–175. IEEE, 2008.

[6] Sara Bernardini, Maria Fox, Derek Long, and Chiara Piacentini. Deterministic versus probabilistic methods for searching for an evasive target. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[7] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299–316, 2011.

[8] Jiri Sgall. Solution of David Gale's lion and man problem. *Theoretical Computer Science*, 259(1):663–670, 2001.

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

[10] Rufus Isaacs. Differential games ii. 1954.

[11] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[12] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, 28(3):357–375, 2009.

[13] Katie Laventall and Jorge Cortés. Coverage control by multi-robot networks with limited-range anisotropic sensory. *International Journal of Control*, 82(6):1113–1121, 2009.

[14] Wei Li and Christos G Cassandras. Distributed cooperative coverage control of sensor networks. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 2542–2547. IEEE, 2005.

[15] Minyi Zhong and Christos G Cassandras. Distributed coverage control and data collection with mobile sensor networks. *Automatic Control, IEEE Transactions on*, 56(10):2445–2455, 2011.

[16] Mac Schwager, James McLurkin, and Daniela Rus. Distributed coverage control with sensory feedback for networked robots. In *robotics: science and systems*, 2006.

[17] Yi Guo and Mohanakrishnan Balakrishnan. Complete coverage control for nonholonomic mobile robots in dynamic environments. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1704–1709. IEEE, 2006.

[18] Noa Agmon, Noam Hazon, Gal Kaminka, et al. Constructing spanning trees for efficient multi-robot coverage. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1698–1703. IEEE, 2006.

[19] Peter F Hokayem, Dušan Stipanović, and Mark W Spong. On persistent coverage control. In *Decision and Control, 2007 46th IEEE Conference on*, pages 6130–6135. IEEE, 2007.

[20] Rene Vidal, Omid Shakernia, H Jin Kim, David Hyunchul Shim, and Shankar Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *Robotics and Automation, IEEE Transactions on*, 18(5):662–669, 2002.

[21] Kamil Dedecius. Diffusion estimation of state-space models: Bayesian formulation. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

[22] Zhijun Tang and Ümit Özgüner. Sensor fusion for target track maintenance with multiple uavs based on bayesian filtering method and hospitability map. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 19–24. IEEE, 2003.

[23] Emrah Adamey and Umit Ozguner. Cooperative multitarget tracking and surveillance with mobile sensing agents: A decentralized approach. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1916–1922. IEEE, 2011.

[24] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *Robotics & Automation Magazine, IEEE*, 13(3):16–25, 2006.

[25] KE Trummel and JR Weisinger. Technical note-the complexity of the optimal searcher path problem. *Operations Research*, 34(2):324–327, 1986.

[26] Timothy H Chung and Joel W Burdick. Analysis of search decision making using probabilistic search strategies. *Robotics, IEEE Transactions on*, 28(1):132–144, 2012.

[27] James N Eagle and James R Yee. An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations research*, 38(1): 110–114, 1990.

[28] Hiroyuki Sato and Johannes O Royset. Path optimization for the resource-constrained searcher. *Naval Research Logistics (NRL)*, 57(5):422–440, 2010.

[29] Timothy H Chung, Moshe Kress, and Johannes O Royset. Probabilistic search optimization and mission assignment for heterogeneous autonomous agents. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 939–945. IEEE, 2009.

[30] Joao P Hespanha, Maria Prandini, and Shankar Sastry. Probabilistic pursuit-evasion games: A one-step nash approach. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2272–2277. IEEE, 2000.

[31] Johannes O Royset and Hiroyuki Sato. Route optimization for multiple searchers. *Naval Research Logistics (NRL)*, 57(8):701–717, 2010.

[32] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[33] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for a class of pursuit-evasion games. In *Algorithmic foundations of robotics IX*, pages 71–87. Springer, 2010.

[34] Adonis Antoniades, H Jin Kim, and Shankar Sastry. Pursuit-evasion strategies for teams of multiple agents with incomplete information. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1, pages 756–761. IEEE, 2003.

[35] Ying-Chun Chen, Huan Qi, and Shan-Shan Wang. Multi-agent pursuit-evasion algorithm based on contract net interaction protocol. In *International Conference on Natural Computation*, pages 482–489. Springer, 2005.

[36] Ying-Chun Chen, Huan Qi, and Xia Liu. Mas-based pursuit-evasion algorithm under unknown environment. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 1, pages 265–269. IEEE, 2005.

[37] Shaunak D Bopardikar, Francesco Bullo, and Joao P Hespanha. On discrete-time pursuit-evasion games with sensing limitations. *IEEE Transactions on Robotics*, 24(6):1429–1439, 2008.

[38] Donald R DelBalzo and KP Hemsteter. Grasp multi-sensor search tactics against evading targets. In *OCEANS'02 MTS/IEEE*, volume 1, pages 54–59. IEEE, 2002.

[39] Hongyang Qu, Andreas Kolling, and Sandor M Veres. Formulating robot pursuit-evasion strategies by model checking. *IFAC Proceedings Volumes*, 47(3):3048–3055, 2014.

[40] Geoffrey Hollinger, Athanasios Kehagias, and Sanjiv Singh. Gsst: Anytime guaranteed search. *Autonomous Robots*, 29(1):99–118, 2010.

[41] Athanasios Kehagias, Geoffrey Hollinger, and Sanjiv Singh. A graph search algorithm for indoor pursuit/evasion. *Mathematical and Computer Modelling*, 50(9): 1305–1317, 2009.

[42] Patrick Vincent and Izhak Rubin. A framework and analysis for cooperative search using uav swarms. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 79–86. ACM, 2004.

[43] Yaniv Altshuler, Vladimir Yanovsky, Israel A Wagner, and Alfred M Bruckstein. Efficient cooperative search of smart targets using uav swarms. *Robotica*, 26(4): 551–557, 2008.

[44] Timothy G McGee and J Karl Hedrick. Guaranteed strategies to search for mobile evaders in the plane. In *American Control Conference, 2006*, pages 6–pp. IEEE, 2006.

[45] Benjamín Tovar and Steven M LaValle. Visibility-based pursuit—evasion with bounded speed. *The International Journal of Robotics Research*, 27(11-12):1350–1360, 2008.

[46] Yue Wang, Islam Hussein, et al. Awareness coverage control over large-scale domains with intermittent communications. *Automatic Control, IEEE Transactions on*, 55(8):1850–1859, 2010.

[47] Cheng Song, Lu Liu, Gang Feng, Yong Wang, and Qing Gao. Persistent awareness coverage control for mobile sensor networks. *Automatica*, 49(6):1867–1873, 2013.

[48] Richard A Wise and Rolf T Rysdyk. Uav coordination for autonomous target tracking. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Keystone, CO, Aug*, pages 21–24, 2006.

[49] Vahram Stepanyan and Naira Hovakimyan. Adaptive disturbance rejection controller for visual tracking of a maneuvering target. *Journal of guidance, control, and dynamics*, 30(4):1090–1106, 2007.

[50] Sonia Martínez and Francesco Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

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

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

[53] Abhijit Sinha, Thiagalingam Kirubarajan, and Yaakov Bar-Shalom. Autonomous ground target tracking by multiple cooperative uavs. In *Aerospace Conference, 2005 IEEE*, pages 1–9. IEEE, 2005.

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

[55] Pantelis Isaiah and Tal Shima. Motion planning algorithms for the dubins travelling salesperson problem. *Automatica*, 53:247–255, 2015.

[56] Ketan Savla, Francesco Bullo, and Emilio Frazzoli. On traveling salesperson problems for dubins' vehicle: stochastic and dynamic environments. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 4530–4535. IEEE, 2005.

[57] Ross P Anderson and Dejan Milutinović. The dubins traveling salesperson problem with stochastic dynamics. In *ASME 2013 Dynamic Systems and Control Conference*, pages V002T20A003–V002T20A003. American Society of Mechanical Engineers, 2013.

[58] Ketan Savla, Emilio Frazzoli, and Francesco Bullo. Traveling salesperson problems for the dubins vehicle. *Automatic Control, IEEE Transactions on*, 53(6):1378–1391, 2008.

[59] Scott A Miller, Zachary A Harris, and Edwin KP Chong. A pomdp framework for coordinated guidance of autonomous uavs for multitarget tracking. *EURASIP Journal on Advances in Signal Processing*, 2009(1):1–17, 2009.

[60] Shankarachary Ragi and Edwin KP Chong. Dynamic uav path planning for multitarget tracking. In *American Control Conference (ACC), 2012*, pages 3845–3850. IEEE, 2012.

[61] Ruijie He, Abraham Bachrach, and Nicholas Roy. Efficient planning under uncertainty for a target-tracking micro-aerial vehicle. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1–8. IEEE, 2010.

[62] Umit Y Ogras, Oguz H Dagci, and Umit Ozguner. Cooperative control of mobile robots for target search. In *Mechatronics, 2004. ICM'04. Proceedings of the IEEE International Conference on*, pages 123–128. IEEE, 2004.

[63] Michel Toulouse, Krishnaiyan Thulasiraman, and Fred Glover. Multi-level cooperative search: A new paradigm for combinatorial optimization and an application to graph partitioning. In *European Conference on Parallel Processing*, pages 533–542. Springer, 1999.

[64] Ivan Maza and Anibal Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007.

[65] Joao V Messias, Matthijs Spaan, and Pedro U Lima. Efficient offline communication policies for factored multiagent pomdps. In *Advances in Neural Information Processing Systems*, pages 1917–1925, 2011.

[66] Stefan Witwicki, Jose Carlos Castillo, Joao Messias, Jesus Capitan, Francisco S Melo, Pedro U Lima, and Manuela Veloso. Autonomous surveillance robots: A decision-making framework for networked muiltiagent systems. *IEEE Robotics & Automation Magazine*, 24(3):52–64, 2017.

[67] Geoffrey Hollinger, Sanjiv Singh, Joseph Djugash, and Athanasios Kehagias. Efficient multi-robot search for a moving target. *The International Journal of Robotics Research*, 28(2):201–219, 2009.

[68] Iadine Chades, Bruno Scherrer, and François Charpillet. A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 57–62. ACM, 2002.

[69] Rosemary Emery-Montemerlo, Geoff Gordon, Jeff Schneider, and Sebastian Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 136–143. IEEE Computer Society, 2004.

[70] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5): 1071–1088, 1973.

[71] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101 (1-2):99–134, 1998.

[72] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[73] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[74] Minlue Wang and Richard Dearden. Run-time improvement of point-based pomdp policies. In *IJCAI*, pages 2408–2414, 2013.

[75] Matthijs TJ Spaan and Nikos Vlassis. Perseus: Randomized point-based value iteration for pomdps. *Journal of artificial intelligence research*, 24:195–220, 2005.

[76] Trey Smith and Reid Simmons. Point-based pomdp algorithms: Improved analysis and implementation. *arXiv preprint arXiv:1207.1412*, 2012.

[77] Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40, 2005.

[78] Eric A Hansen. Solving pomdps by searching in policy space. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 211–219. Morgan Kaufmann Publishers Inc., 1998.

[79] Pascal Poupart and Craig Boutilier. Vdcbpi: an approximate scalable algorithm for large pomdps. In *Advances in Neural Information Processing Systems*, pages 1081–1088, 2005.

[80] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94, 2000.

[81] Stéphane Ross, Joëlle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32: 663–704, 2008. ISSN 10769757. doi: 10.1613/jair.2567.

[82] Trey Smith and Reid Simmons. *Probabilistic planning for robotic exploration*. PhD thesis, Carnegie Mellon University, The Robotics Institute, 2007.

[83] Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-Draa. An online pomdp algorithm for complex multiagent environments. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 970–977. ACM, 2005.

[84] Sébastien Paquet, Brahim Chaib-draa, and Stéphane Ross. Hybrid pomdp algorithms. In *Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM-06)*, pages 133–147, 2006.

[85] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[86] EKP Chong, C Kreucher, and AO Hero III. Pomdp approximation methods based on heuristics and simulation. *Foundations and Applications of Sensor Management*, 8:95–120, 2007.

[87] Eric A Hansen, Daniel S Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715, 2004.

[88] John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[89] Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[90] Daniel Szer, François Charpillet, and Shlomo Zilberstein. Maa*: A heuristic search algorithm for solving decentralized pomdps. *arXiv preprint arXiv:1207.1359*, 2012.

[91] Camille Besse and Brahim Chaib-draa. Parallel rollout for online solution of dec-pomdps. In *FLAIRS Conference*, pages 619–624, 2008.

[92] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Multi-agent online planning with communication. In *ICAPS*, 2009.

[93] Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[94] Rosemary Emery-Montemerlo. *Game-Theoretic Control for Robot Teams.* PhD thesis, Rutgers, The State University of New Jersey, 2005.

[95] Martin L Puterman. Markov decision processes. discrete stochastic dynamic programming mvspa. 2005.

[96] Hyeong Soo Chang, Robert Givan, and Edwin KP Chong. Parallel rollout for online solution of partially observable markov decision processes. *Discrete Event Dynamic Systems*, 14(3):309–341, 2004.

[97] Richard Dearden and Craig Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1-2):219–283, 1997.

[98] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[99] Edwin KP Chong, Christopher M Kreucher, and Alfred O Hero III. Pomdp approximation using simulation and heuristics. In *Foundations and Applications of Sensor Management*, pages 95–119. Springer, 2008.

[100] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Malaysia; Pearson Education Limited,, 2016.

[101] Songhwai Oh, Shankar Sastry, and Luca Schenato. A hierarchical multiple-target tracking algorithm for sensor networks. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2197–2202. IEEE, 2005.

[102] Przemek. anneal.m. `https://github.com/adgon92/optimalization-project/blob/dd04eafd18d8cc0adb87f880e2421947c2f053e0/examples/anneal.m#L1-L122`, 2015.

[103] PJ van Laarhoven and EH Aarts. *Simulated Annealing: Theory and Applications*, volume 37. Springer Science & Business Media, 2013.

[104] Farrokh Janabi-Sharifi and D Vinke. Integration of the artificial potential field approach with simulated annealing for robot path planning. In *Intelligent Control, 1993., Proceedings of the 1993 IEEE International Symposium on*, pages 536–541. IEEE, 1993.

[105] Min Gyu Park, Jae Hyun Jeon, and Min Cheol Lee. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on*, volume 3, pages 1530–1535. IEEE, 2001.

[106] Qidan Zhu, Yongjie Yan, and Zhuoyi Xing. Robot path planning based on artificial potential field approach with simulated annealing. In *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, volume 2, pages 622–627. IEEE, 2006.