DISSERTATION

COOPERATIVE CONTROL OF MOBILE SENSOR PLATFORMS IN DYNAMIC ENVIRONMENTS

Submitted by

Shankarachary Ragi

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2014

Doctoral Committee:

Advisor: Edwin K. P. Chong

Diego Krapf
J. Rockey Luo
Juliana Oprea

ABSTRACT

COOPERATIVE CONTROL OF MOBILE SENSOR PLATFORMS IN DYNAMIC ENVIRONMENTS

We develop guidance algorithms to control mobile sensor platforms, for both centralized and decentralized settings, in dynamic environments for various applications. More precisely, we develop control algorithms for the following mobile sensor platforms: unmanned aerial vehicles (UAVs) with on-board sensors for multitarget tracking, autonomous amphibious vehicles for flood-rescue operations, and directional sensors (e.g., surveillance cameras) for maximizing an information-gain-based objective function. The following is a brief description of each of the above-mentioned guidance control algorithms.

We develop both centralized and decentralized control algorithms for UAVs based on the theories of partially observable Markov decision process (POMDP) and decentralized POMDP (Dec-POMDP) respectively. Both POMDPs and Dec-POMDPs are intractable to solve exactly; therefore we adopt an approximation method called nominal belief-state optimization (NBO) to solve (approximately) the control problems posed as a POMDP or a Dec-POMDP.

We then address an amphibious vehicle guidance problem for a flood rescue application. Here, the goal is to control multiple autonomous amphibious vehicles while minimizing the average rescue time of multiple human targets stranded in a flood situation. We again pose this problem as a POMDP, and extend the above-mentioned NBO approximation method to solve the guidance problem.

In the final phase, we study the problem of controlling multiple 2-D directional sensors while maximizing an objective function based on the information gain corresponding to multiple target locations. This problem is found to be a combinatorial optimization problem, so we develop heuristic methods to solve the problem approximately, and provide analytical

results on performance guarantees. We then improve the performance of our heuristics by applying an approximate dynamic programming approach called rollout.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Dr. Edwin K. P. Chong for his support and guidance throughout my stay at Colorado State University (CSU), for giving me the opportunity to work on a variety of challenging research problems, and for giving me the freedom in choosing the research problems of my interest. I would also like to thank Dr. Diego Krapf, Dr. Rockey Luo, and Dr. Juliana Oprea for serving as my doctoral committee members.

I am forever indebted to my loving parents, Ragi Venkatachary and Ragi Venkata Ramanamma, for their constant support and encouragement throughout my life. I would like to extend my appreciation to my sister Ragi Kalpana for sharing my childhood experiences and memories, and for filling my childhood with joy.

I would like to thank my friends and colleagues at CSU, Chintan Joshi, Thiyagarajan Chockalingam, Sahil Mehta, Manish Mohanpurkar, Kaustubh Gadkari, Athul S. Chandran, Arunachalam Lakshminarayanan, Yajing Liu, Wenbing Dang, Ramin Zahedi, Yang Zhang, and Zhenliang Zhang, for accompanying me through my stay at CSU. My regards also go to my friends from IIT Madras (where I did my undergraduate studies) Anuprem, Karthik, Jyothi, Vasanth, and Vikas who visited my place in Fort Collins during my university holidays and made these holidays completely fun-filled.

I would like to acknowledge the following agencies for supporting my research on UAV guidance (covered in Chapter 4): Northrop Grumman Corporation (via RMATI program) and AFOSR (via contract FA9550-09-1-0518). I would also like to acknowledge Fulbright Foundation for supporting my research on AAV guidance (covered in Chapter 6). I would like to acknowledge my collaborators Dr. ChingSeong Tan (a faculty member at Multimedia University in Malaysia) and Dr. Hans D. Mittelmann (a faculty member from the School of Mathematical and Statistical Sciences at Arizona State University) for their insightful discussions and valuable suggestions during my research.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this study, a mobile sensor platform is a system with the following characteristics: 1) senses the environment, e.g., via optical sensors, radar, and sonar; 2) communicates with other mobile sensor platforms or a base station; 3) runs computations via an on-board computer; 4) has controllable aspects, e.g., sensing direction, location, and waveform. Per the above definition, the following systems can be considered as mobile sensor platforms: any autonomous vehicle with on-board sensors (e.g., unmanned aerial vehicles, autonomous amphibious vehicles, unmanned underwater vehicles), and directional sensors (e.g., PTZ camera). The cooperative control of mobile sensor platforms is an active area of research with increasing focus on both defence and civilian applications like crop monitoring, surveillance, tracking, convoy protection, search and rescue operations (e.g., flood rescue support and fire-fighting support), and wildfire suppression. The autonomy for mobile sensor platforms, especially for the examples of mobile sensors listed before, is becoming increasingly important because of the nature of missions, where the missions are dull, dirty, and dangerous for humans. With this motivation, we develop guidance control methods for the following applications:

- Controlling multiple unmanned aerial vehicles (UAVs) for tracking multiple targets.
- Controlling multiple autonomous amphibious vehicles (AAVs) for flood rescue support.
- Controlling directional sensors for maximizing an information-gain-based objective function.

In the first phase (Chapters 4 and 5), we develop guidance control methods for UAVs tracking multiple targets in both centralized and decentralized settings. In the centralized

setting, we assume that there is a notional fusion center, which collects measurements from the sensors, applies data fusion on the observations, maintains and updates the posterior distribution on the system-state variables, computes a control plan for the mobile sensor platforms to maximize (or minimize) a certain reward (or cost). In this setting, we develop the UAV guidance method based on the theory of *partially observable Markov decision process* (POMDP). A POMDP is a mathematical framework useful for solving resource control problems. It is intractable to solve a POMDP exactly. Therefore, the literature on POMDPs has focused on approximation methods [1]. In our study, we adopt an approximation method called *nominal belief-state optimization* (NBO) to solve our UAV guidance control problem posed as a POMDP. A detailed description of POMDP is covered in Chapter 2. We then extend this algorithm to incorporate wind disturbance on UAVs, collision avoidance (among UAVs, and between UAVs and obstacles), threat evasion, evasive-target tracking, and tracks-swap avoidance. In the decentralized setting, there is no fusion center (or central controller), and each UAV is an independent decision maker. In this setting, we develop a decentralized control method for UAVs, while inducing coordination among the UAVs by allowing communication between the UAVs, and this communication comes at a cost. Here, the development of our decentralized guidance control algorithm is based on the theory of *decentralized partially observable Markov decision process* (Dec-POMDP). A Dec-POMDP, like a POMDP, is hard to be solved exactly. Therefore, we extend the above-mentioned NBO method to solve our UAV guidance problem posed a Dec-POMDP. A detailed description of Dec-POMDP is provided in Chapter 3.

In the second phase (Chapter 6), we study the problem of controlling autonomous amphibious vehicles (AAVs) for flood rescue operations. More precisely, we develop guidance control algorithms for AAVs to minimize the average rescue time of human targets stranded in a flood situation. We pose this problem as a POMDP, and we extend the above-mentioned NBO method to solve this guidance problem. We then compare the performance of this algorithm with a "greedy" approach.

In the third and final phase (Chapter 7), we study the problem of controlling directional sensors for maximizing the information gain corresponding to multiple targets. This is a combinatorial optimization problem, and the computation time increases exponentially with the number of sensors. Therefore, we develop polynomial-time heuristic approaches to solve this problem approximately. We then apply an approximate-dynamic-programming approach called rollout on our heuristics to improve the performance of the heuristics. We then compare the performance of these heuristic approaches analytically and empirically.

CHAPTER 2

CENTRALIZED GUIDANCE CONTROL FRAMEWORK

2.1 Partially Observable Markov Decision Process

Partially observable Markov decision process (POMDP) [1] is a mathematical framework useful for solving resource control problems. A POMDP can also be viewed as a hidden Markov reward process. In general, a POMDP is hard to solve exactly. Therefore, the literature on POMDP methods has focused on approximation methods [1]. A POMDP evolves in discrete time-steps; in this study we assume that the length of each time-step is T seconds. We use k as the discrete-time index. The following are the key components of a POMDP:

States. The states are the features of the system that possibly evolve over time and are relevant to the problem of interest. Let $x_k \in X$ represent the state of the system at time k, where X be the set of all possible states.

Actions. The actions are the controllable aspects of the system, i.e., the transition of the system from the current state to the next state depends on the current actions. Let $a_k \in A$ represent the actions at time k, where A is the set of all possible actions.

State-Transition Law. The state-transition law defines the conditional probability distribution over the next state x_{k+1} given the current state x_k and the current action a_k, i.e.,

$$x_{k+1} \sim p_k(\cdot|x_k, a_k),$$

where p_k represents a conditional probability distribution over the state space X.

Observations and Observation-Law. Let $z_k \in Z$ be the observation at time k, where Z is the observation space. The observation law specifies the condition distribution over the observation space Z given the current state x_k and possibly the current action a_k, i.e.,

$$z_k \sim q_k(\cdot|x_k, a_k),$$

where q_k represents a conditional probability distribution over the observation space Z.

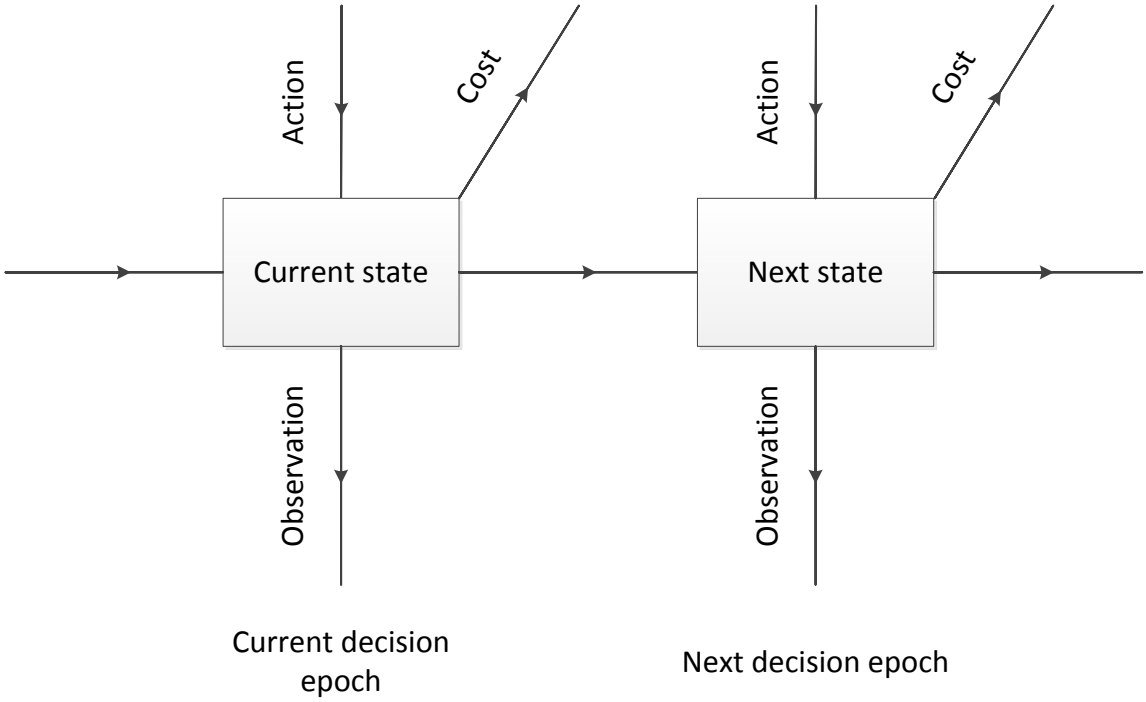
Cost Function. The cost function at time k represents the cost (a real number) of taking an action a_k given the current state x_k. Let $C_k : X \times A \to \mathbb{R}$ represent the cost function at time k.

Belief State. The belief-state at any given time is the posterior distribution over the state space X given the history of observations and actions. Let $b_k \in B$ represent the belief-state at time k, where B is the set of all distributions over the state space X.

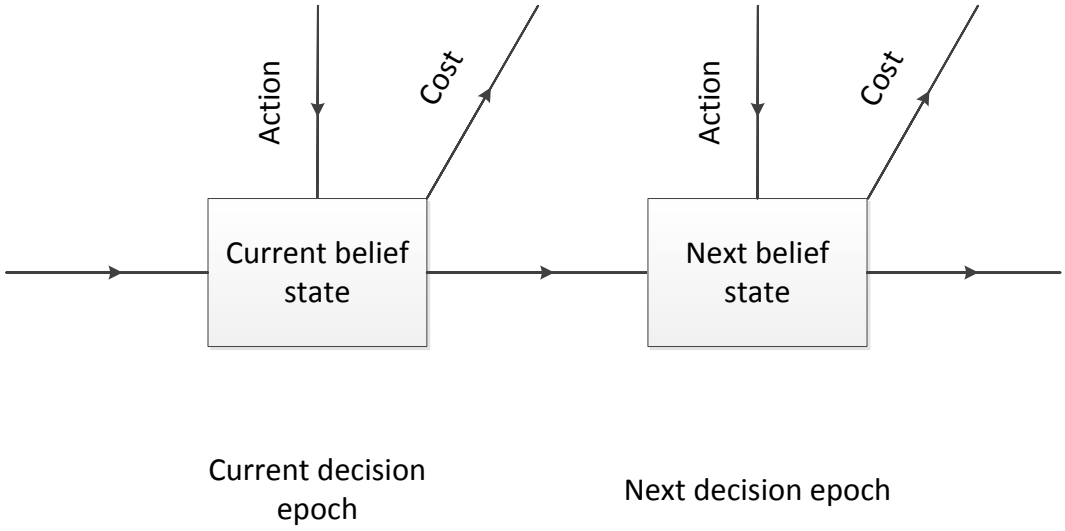
Figure 2.1(a) shows a symbolic representation of the POMDP process. The POMDP process begins at time $k = 0$ at a (random) initial state. As the process evolves, the state transitions to a (random) next state from the current state according to the state-transition law given the action. An action taken at a given state incurs a cost, which is given by the cost function. At every time-step, the system generates an observation, which depends on the current state and action. At every time-step, this observation is used to infer the actual underlying state. However, there will be some uncertainty in the knowledge of the underlying state; this uncertainty is represented by the *belief state*. The belief state is the posterior distribution over the underlying state, which is updated according to the Bayes' rule. A POMDP can be viewed as a fully-observable Markov decision process (MDP) with state space B as shown in Figure 2.1(b).

2.1.1 Optimization Objective

A POMDP problem boils down to an optimization problem, where the objective is to find actions over a time horizon H such that the expected cumulative cost is minimized. The expected cumulative cost, to be minimized over the action sequence $u_0, u_1, \ldots, u_{H-1}$, is



(a) POMDP with state space $\mathcal{X}$.



(b) Fully observable MDP with state space $\mathcal{B}$.

Figure 2.1: Symbolic representations.

given by:

$$J_H = \mathrm{E}\left[\sum_{k=0}^{H-1} C(x_k, u_k)\right]. \tag{2.1}$$

The action chosen at time k should be allowed to depend on the history of all observable quantities till time $k-1$. If an optimal choice of such actions exist, then there exists an optimal sequence of actions that depend only on the "belief-state feedback" [1]. Therefore, the objective function J_H can be written in terms of the belief states as follows:

$$J_H = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k, u_k) \,\middle|\, b_0\right], \tag{2.2}$$

where $c(b_k, u_k) = \int C(x, u_k) b_k(x)\,\mathrm{d}x$ and b_0 is the given initial belief state.

2.1.2 Optimal Policy

Given the optimization problem, the goal is to find, at each time k, the optimal policy $\pi_k^* : \mathcal{B} \to \mathcal{U}$ such that if the action $u_k = \pi_k^*(b_k)$ is performed at time k, the objective function (2.2) is minimized. According to the Bellman's principle of optimality [2], the optimal objective function value can be written in the following form:

$$J_H^*(b_0) = \min_u \left\{ c(b_0, u) + \mathrm{E}\left[J_{H-1}^*(b_1) \,\middle|\, b_0, u \right] \right\}, \tag{2.3}$$

where b_1 is the random next belief state, J_{H-1}^* is the optimal cumulative cost over the horizon $H-1 : k = 1, 2, \ldots, H-1$, and $\mathrm{E}[\cdot|b_0, u]$ is the conditional expectation given the current belief state b_0 and an action u taken at time $k = 0$. The Q-value of taking an action u given the current belief state b_0 is as follows:

$$Q_H(b_0, u) = c(b_0, u) + \mathrm{E}\left[J_{H-1}^*(b_1) \,\middle|\, b_0, u \right].$$

The optimal policy (from Bellman's principle) at time $k = 0$ is given by:

$$\pi_0^*(b_0) = \arg\min_u \, Q_H(b_0, u). \tag{2.4}$$

The optimal policy at time k is given by:

$$\pi_k^*(b_k) = \arg\min_u \, Q_{H-k}(b_k, u). \tag{2.5}$$

We assume a long horizon, which makes the dependence on the horizon of the ECTG negligible, and the optimal policy stationary. Therefore, the stationary optimal policy at time k is given by

$$\pi^*(b_k) = \arg\min_u \; Q(b_k, u)$$

where

$$Q(b_k, u) = c(b_k, u) + \mathrm{E}\left[J_H^*(b_{k+1}) \,|\, b_k, u\right]. \tag{2.6}$$

In practice, the second term in the Q value (6.3) is hard to obtain exactly. There are several methods in the literature to obtain the Q-value approximately, which include: heuristic ECTG [3], parametric approximation [4,5], policy rollout [6], hindsight optimization [7, 8], foresight optimization [9].

CHAPTER 3

DECENTRALIZED GUIDANCE CONTROL FRAMEWORK

3.1 Decentralized Partially Observable Markov Decision Process

Decentralized partially observable Markov decision process (Dec-POMDP for short) [10] is a mathematical framework useful for modeling multi-agent resource control problems where the decision making is decentralized. The Dec-POMDP approach has the following advantages: 1) it offers a general approach to dealing with resource management in a multi-agent scenario with decentralized control, 2) it is a non-myopic approach, which trades off short-term for long-term performance, 3) in comparison to the existing greedy/myopic approaches in the literature, the Dec-POMDP approach results in more effective exploitation of limited resources in a decentralized setting. In addition, we explicitly optimize the communication between the agents. In general, solving a Dec-POMDP exactly is intractable. We can extend centralized POMDP (*partially observable Markov decision process*) approximation methods to solve the Dec-POMDP. The Dec-POMDP evolves in discrete time-steps, where the length of each time-step is T seconds. We use k as the discrete-time index. For the purpose of this study, we assume that the decision epochs at each agent are synchronized. In this study, an agent is an autonomous system that has sensing, computational, and communication abilities.

In this study, we assume that there is an underlying communication network over which the agents can communicate information to each other. We will not concern ourselves with the details of this underlying network. Instead, for each pair of agents, we will treat the

end-to-end path through the network between them as an abstract *communication link* over which the two agents can exchange information. To account for the cost of communicating information over the network, we assign a numerical value to each communication link (which we will call the *cost* of the link). For example, we can define the cost of a link as a value that is proportional to the distance between the two associated agents. The following are the key components of a Dec-POMDP.

Agents. We assume that there are N agents in the system. Let $\mathcal{I} = \{1, \ldots, N\}$ represent the set of agents.

States. The states are the features of the system that possibly evolve over time and are relevant to the problem of interest, i.e., the state of each agent, the state of the environment, and other relevant features of the system. Let $x_k \in X$ represent the state of the system at time k, where X be the set of all possible states.

Joint Actions. The joint-action is a tuple, the components of which are actions corresponding to individual agents. Let $u_k = (u_k^1, \ldots, u_k^N)$ represent the joint-action, where u_k^i is the action-vector at agent i. The local action at any agent may include control commands that may influence the state transition of the local agent, and communication decisions like whom to communicate with, when to communicate, and what to communicate.

State Transition Law. The state transition law specifies the next-state distribution given the current state and joint-action, i.e., $x_{k+1} \sim p_k(\cdot|x_k, u_k)$, where p_k is a conditional probability distribution.

Observations and Observation Law. Each joint-observation is a tuple, the components of which are observations generated by individual agents via on-board sensors. Let $z_k = (z_k^1, \ldots, z_k^N)$ be the joint-observation vector at time k, where z_k^i represents the observation at agent i. The observation law specifies the distribution of joint-observations given the current state and joint-action, i.e., $z_k \sim q_k(\cdot|x_k, u_k)$, where q_k is a conditional probability distribution. The observations at each agent can be used to infer the underlying state, albeit with some uncertainty.

Cost Function. A cost function $C(x_k, u_k)$ specifies the cost (real number) of being in a given state x_k and performing a joint-action u_k. The cost function may include a performance measure (e.g., tracking error in a tracking application) and the cost of communication.

The Dec-POMDP starts at a random initial state x_0 (whose distribution is given), and at any typical time-step k, the state x_k transitions to x_{k+1} given the joint-action vector u_k. The joint-action u_k performed at the current state x_k incurs a global cost $C(x_k, u_k)$. As a Dec-POMDP evolves over time as a dynamical process, the agents may not know the underlying state exactly, but each agent generates observations of the underlying state, providing the agent with clues of the actual underlying states. Given the Dec-POMDP formulation, the goal is to find joint-actions over a horizon H such that the expected cumulative cost, over a time horizon H, is minimized.

An agent may not know the action taken and the observation generated at another agent. An agent may decide to communicate with another agent, and these decisions to communicate are embedded into the joint-action vector u_k. The communication among the agents incurs a cost, which is embedded in the global cost function $C(x_k, u_k)$. The local observations allow each agent to infer, with some uncertainty, what states actually occurred. This uncertainty is represented by the *local belief-state*, which is the *a posteriori* distribution of the underlying state given the history of local observations and local actions made by that agent, including the information gathered from other agents. Just as in centralized POMDPs, in the decentralized case the local belief-state will be used as "feedback" information that is needed for controlling the system. In other words, we seek an optimal joint-policy that depends only on the local belief-states.

3.1.1 Objective and Optimal Policy

The problem is to minimize the cumulative cost over horizon H, given by

$$\mathrm{E}\left[\sum_{k=0}^{H-1} C(x_k, u_k)\right].$$

In the centralized case (if it was a POMDP problem), this objective function can be written in terms of "global" belief-states:

$$J(b_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k, u_k) \middle| b_0\right],$$

where $c(b,u) = \int C(x,u)b(x)\,dx$, b_k is the "global" belief-state, i.e., the posterior distribution at time k, and $\mathrm{E}\left[\cdot|b_0\right]$ represents conditional expectation given the initial belief-state b_0 at time $k = 0$. The goal is to pick the joint-actions over time so that the objective function is minimized. In general, the actions chosen for agents at each time should be allowed to depend on the entire history of observations and actions up to that time. However, if an optimal choice of such a sequence of actions exists, then there is an optimal choice of actions that depends only on "belief-state feedback." Hence, if we ignore the decentralized nature of the problem, what we seek is an optimal *policy*, which maps the belief-state at each time to the joint-action tuple at that time. The optimal policy is characterized by *Bellman's principle* [2], according to which the optimal action at time k is

$$\pi^*(b_0) = \arg\min_u \left\{c(b_0,u) + \mathrm{E}\left[J^*(b_1)|b_0,u\right]\right\},$$

where b_0 is the initial belief-state, b_1 is the random next belief-state, and $\mathrm{E}[J^*(b_1)|b_0,u]$ is the expected future cost of action u, which is also called the *expected cost-to-go* (ECTG). We assume a long horizon, which makes the dependence on the horizon of the ECTG negligible, and the optimal policy stationary.

Let us define the *Q-value* of taking action u at belief-state b as follows:

$$Q(b,u) = c(b,u) + \mathrm{E}\left[J^*(b')|b,u\right], \tag{3.1}$$

where b' is the random next belief-state. Therefore, the optimal policy is given by

$$\pi^*(b_0) = \arg\min_u Q(b_0,u).$$

In the decentralized case, we do not have access to the "global" belief-state. Instead, every agent maintains a *local* belief-state, which may vary from agent to agent (because the

observation histories differ from agent to agent). Our approach is for each agent to compute its own local action as follows. The agent i computes, at time k,

$$\pi^i(b_k^i) = \arg\min_u Q(b_k^i, u), \tag{3.2}$$

where b_k^i is the local belief-state at agent i at time k. In other words, an agent i computes a joint-action by taking into account only its local belief-state. After the computation of the joint-action, agent i implements its local component. Our approach maintains the *look-ahead* (non-myopic) property that is a common theme among POMDP solution approaches, allowing us to account for the future impact of actions in our decision making.

In practice, it is intractable to compute the Q-value. Therefore, the literature on POMDP methods has focused on approximation methods [1]. In Chapter 5, we extend one such method called *nominal belief-state optimization* (NBO) to solve a UAV guidance problem posed as a Dec-POMDP.

In the next three chapters, we present two applications of POMDP and an application of Dec-POMDP, where we develop guidance control methods for—UAVs for target tracking in centralized and decentralized settings, and autonomous amphibious vehicle guidance for flood rescue support.

CHAPTER 4

UAV GUIDANCE FOR TRACKING MULTIPLE TARGETS VIA POMDP

4.1 Introduction

This chapter presents a path-planning algorithm to guide autonomous unmanned aerial vehicles (UAVs) for tracking multiple ground targets based on the theory of *partially observable Markov decision processes* (POMDPs) [1,11]. The algorithm collects measurements from the sensors (mounted on the UAVs), constructs the tracks, and computes the control commands for the UAVs. The focus of this chapter is to show how to guide UAVs with control variables: forward acceleration and bank angle subject to constraints, account for wind disturbance on UAVs, avoid collisions between UAVs and obstacles and among UAVs, guide UAVs to track targets while evading threats, guide UAVs for tracking evasive targets, and guide UAVs for tracking targets while mitigating track swaps.

The literature is replete with path-planning algorithms for autonomous vehicles; a general survey of these algorithms can be found in [12]. What makes our path-planning method for UAVs different and worth studying is the combination of the following features naturally within the POMDP framework: 1) Our method involves real-time calculation of guidance commands in response to feedback information. 2) Our method has a "look-ahead" property, in that the guidance control at each time-step takes into account the impact of the control over a time horizon into the future. 3) Our method incorporates explicit dynamic constraints on the UAV motion. 4) Our cost criterion is based on tracking error.

Although these features are not novel in their own right, the contribution of this chapter is to combine them naturally within the POMDP framework. For example, the path-planning

algorithms presented in [13–18] also incorporate "look-ahead" and can be implemented in real-time. However, only [14, 17] incorporate the dynamic constraints on UAV motion and only [15] incorporates the tracking error as a cost criterion. Our POMDP method is an integrated approach where all the above features are fused in a single framework—the guidance commands are calculated in real-time (with look-ahead) while incorporating dynamic constraints on UAV motion in response to feedback information, where the cost criterion is tracking error. There are many other methods in the literature that are not designed specifically for UAVs but are designed for autonomous robots (mostly ground-based) in general; e.g., [19–22]. In the recent past, POMDPs have been used to design path-planning algorithms for UAVs. The approach in [23] is similar to that of ours in that the authors used POMDP for designing a UAV guidance algorithm for target tracking and also considered collision avoidance, but differs from our approach in that the authors did not incorporate the dynamic constraints on UAV motion. There are also methods in the recent literature (e.g., [24,25]) where POMDPs are used to develop path-planning algorithms for autonomous robots in general, where the focus was not on UAVs. There are also many schemes in the literature for wind compensation (e.g., [26, 27]) and collision avoidance (e.g., [28–31]). Our study differs from these in that we show how to account for wind compensation and collision avoidance naturally in the context of our POMDP framework. The authors in [32] posed collision avoidance problem as a POMDP, but differs from our method in that: 1) they used a simplified motion model for an aircraft, where the aircraft can maneuver only in the vertical direction and 2) their approach did not involve target tracking.

In practice, POMDP problems are intractable to solve exactly. There are several approximation methods in the literature to solve a POMDP [1]. In this study, we use an approximation method called the *nominal belief-state optimization*(NBO) [11], which is computationally efficient compared to the above mentioned methods. This chapter shows how a variety of features of interest in UAV guidance for target tracking can be incorporated

naturally into the POMDP framework by plugging in the appropriate components into the framework.

4.2 Problem Specification

The targets move on a 2-D plane on the ground. We use a simplified UAV motion model, where the altitude of a UAV is assumed to be constant. The 2-D position coordinates of each UAV are varied by applying the controls—forward acceleration and bank angle. The UAVs are mounted with sensors that generate the position measurements of the targets. These measurements have random errors that are spatially varying, i.e., the measurement error covariance of a target depends on the locations of sensors (UAVs) and the target. A UAV is controlled by forward acceleration (which controls the speed) and bank angle (which controls the heading angle). The values of these control variables are restricted to lie within certain minimum and maximum limits. We model obstacles as regions of various shapes (e.g., circles, rectangles) in the surveillance region that block the path of UAVs and also act as occlusions. For the purpose of this study, a threat is a ground vehicle that actively pursues UAVs that are tracking it. An evasive target is a ground vehicle that actively moves away from UAVs to avoid being tracked. The objective is to minimize the mean-squared error between the tracks and the targets. In Sections 4.5, 4.6, and 4.8, we modify the tracking objectives to avoid collisions between UAVs and obstacles and among UAVs, evade threats, and mitigate track swaps.

4.3 POMDP Formulation and the NBO Approximation Method

A POMDP (*partially observable Markov decision process*) is a controlled dynamical process in discrete time useful for modeling resource control problems (e.g., our guidance problem). A POMDP can also be interpreted as the controlled version of a hidden Markov reward

process. The following subsection defines the key components of POMDP with respect to our guidance problem.

4.3.1 POMDP Ingredients

States. The POMDP states represent those features in the system that possibly evolve over time. We define three sub-systems: the sensors, the targets, and the tracker. Therefore, the state at time k is given by $x_k = (s_k, \chi_k, \xi_k, \mathbf{P}_k)$, where s_k represents the sensor state, χ_k represents the target state, and $(\xi_k, \mathbf{P}_k)$ represents the tracker state. The sensor state includes the locations and velocities of the UAVs and the target state includes the locations, velocities, and accelerations of the targets. The tracker state is a standard in the Kalman filter [33,34], where ξ_k is the posterior mean vector and $\mathbf{P}_k$ is the posterior covariance matrix.

Actions. The actions are the controllable aspects of the system. In this problem, the actions include the forward acceleration and the bank angle of each UAV. More precisely, the action at time k is given by $u_k = (a_k, \phi_k)$, where a_k and ϕ_k are vectors containing the forward acceleration and bank angle respectively for each UAV.

Observations and Observation Law. The POMDP states are not fully observable; only a random observation of the underlying state is available at any given time. Let χ_k^{pos} and s_k^{pos} be the position vectors of a target and a sensor/UAV respectively. Then the observation of the target's position is given by

$$z_k^{\chi} = \begin{cases} \chi_k^{\text{pos}} + w_k & \text{if target} \\ & \text{is visible,} \\ \text{no measurement} & \text{otherwise,} \end{cases} \tag{4.1}$$

where w_k represents a random measurement error whose distribution depends on the locations of the UAV (s_k^{pos}) and the target (χ_k^{pos}). The sensor and the tracker states are assumed to be fully observable.

State-Transition Law. The state-transition law specifies the next-state distribution given an action taken at a current state. Since we defined three sub-systems, it is convenient

to define the state-transition law for each sub-system separately. The sensor state evolves according to $s_{k+1} = \psi(s_k, u_k)$, where ψ is a mapping function (defined later). The target state evolves according to $\chi_{k+1} = f(\chi_k) + v_k$, where v_k represents an i.i.d. random sequence and f represents the target motion model (also defined later). Finally, the tracker state evolves according to the Kalman filter equations with a data association technique called *joint probabilistic data association* (JPDA) [33, 35]. When the target observations are not available (when occluded), only the prediction step in the Kalman filter is performed and the update equation is not performed.

Cost Function. The cost function specifies the cost of taking an action in a given state. We use the mean-squared error between the tracks and the targets as the cost function:

$$C(x_k, u_k) = \mathrm{E}_{v_k, w_{k+1}} \left[||\chi_{k+1} - \xi_{k+1}||^2 \,\middle|\, x_k, u_k \right].$$

Belief State. The belief state is the posterior distribution of the underlying state, which is updated incrementally using Bayes rule given the observations. The belief state at time k is given by $b_k = (b_k^s, b_k^\chi, b_k^\xi, b_k^{\mathbf{P}})$, where $b_k^s = \delta(s - s_k)$, $b_k^\xi = \delta(\xi - \xi_k)$, $b_k^{\mathbf{P}} = \delta(\mathbf{P} - \mathbf{P}_k)$ (since the sensor and the tracker states are fully observable), and b_k^χ is the posterior distribution of the target state.

4.3.2 Optimal Policy

Given the POMDP formulation, the objective is to choose actions over a time horizon H, $k = 0, 1, \ldots, H-1$, such that the expected cumulative cost is minimized. The expected cumulative cost over the time horizon H can be written as:

$$J_H = \mathrm{E}\left[\sum_{k=0}^{H-1} C(x_k, u_k) \right].$$

The action chosen at time k should be allowed to depend on the history of all observable quantities till time $k-1$. If an optimal choice of such actions exist, then there exists an optimal sequence of actions that depends only on "belief-state feedback" [1]. Therefore, the

objective function can be written in terms of the belief states as follows:

$$J_H = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k, u_k) \,\middle|\, b_0\right],$$

where $c(b_k, u_k) = \int C(x, u_k) b_k(x)\,\mathrm{d}x$.

As discussed in Chapter 2, the optimal policy from Bellman's principle [2] is given by

$$\pi_0^*(b_0) = \arg\min_u Q_H(b_0, u),$$

where the "Q-value" of taking an action u is given by

$$Q_H(b_0, u) = c(b_0, u) + \mathrm{E}\left[J_{H-1}^*(b_1) \,\middle|\, b_0, u\right].$$

In practice, the second term in the Q-value (above) is hard to obtain exactly. A number of methods have been studied in the literature [1,3–9] to approximate the Q-value. We use one such approximation method called the *nominal belief-state optimization* (NBO), which was introduced in [11] along with other approximations and techniques specific to our guidance problem.

4.3.3 NBO Approximation Method

Although there are several approximation methods in the literature to solve POMDPs, we use the NBO method because it is computationally less burdensome than other POMDP approximation methods. In practice, a UAV guidance algorithm should be implementable in real-time, which requires an approach that is not computationally burdensome. Therefore, we choose NBO approximation method, which is computationally efficient. The algorithm run-time in MATLAB for NBO method is approximately 350 milliseconds on a lab computer (Intel Core i7-860 Quad-Core Processor (8MB Cache, 2.80 GHz)). The algorithm run-time can be greatly reduced on a better processor and also by optimizing the code.

Let us assume there are N_{targs} targets. We can represent the target state as $\chi_k = (\chi_k^1, \chi_k^2, \ldots, \chi_k^{N_{\text{targs}}})$, where χ_k^i represents the state of the ith target. Let the track state be

$\xi_k = (\xi_k^1, \ldots, \xi_k^{N_{\text{targs}}})$ and $\mathbf{P}_k = (\mathbf{P}_k^1, \ldots, \mathbf{P}_k^{N_{\text{targs}}})$, where $(\xi_k^i,\ \mathbf{P}_k^i)$ is the track state corresponding to the ith target. We use a linearized target motion model with zero-mean noise to model the target-state dynamics, as given below $(\forall i)$:

$$\chi_{k+1}^i = \mathbf{F}_k \chi_k^i + v_k^i,\ v_k^i \sim \mathcal{N}(0, \mathbf{Q}_k),\ i \in \{1, \ldots, N_{\text{targs}}\} \tag{4.2}$$

and the observations (at any UAV) are as follows:

$$z_k^{\chi^i} = \begin{cases} \mathbf{H}_k \chi_k^i + w_k^i & \text{if target} \\ & \text{is visible,} \\ \text{no measurement} & \text{otherwise,} \end{cases}$$

where $w_k^i \sim \mathcal{N}(0, \mathbf{R}_k(\chi_k^i, s_k))$, $\mathbf{F}_k$ is the target motion model (same for all targets), and $\mathbf{H}_k$ is the observation model (4.1) (same for every target) according to which only the position of a target is observed. The state of the ith target (χ_k^i) includes its 2-D position coordinates (x_k, y_k), its velocities (v_k^x, v_k^y) and accelerations (a_k^x, a_k^y) in x and y directions, i.e, $\chi_k^i = [x_k, y_k, v_k^x, v_k^y, a_k^x, a_k^y]^{\mathrm{T}}$. Therefore, the observation model is of the form $\mathbf{H}_k = [\mathbf{I}_{2\times 2}, \mathbf{0}_{4\times 4}]$. We adopt the *constant velocity* (CV) model [33, 34] for target dynamics (4.2), which defines $\mathbf{F}_k$. Since we assumed Gaussian distributions, the belief state corresponding to the ith target can be expressed (or approximated) as:

$$b_k^{\chi^i}(\chi) = \mathcal{N}\left(\chi - \xi_k^i, \mathbf{P}_k^i\right),$$

where ξ_k^i and $\mathbf{P}_k^i$ are the track states corresponding to the ith target, which evolve according to the JPDA algorithm [33, 35].

According to the NBO method, the objective function can be approximated as follows:

$$J_H(b_0) \approx \sum_{k=0}^{H-1} c(\hat{b}_k, u_k),$$

where $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_{H-1}$ is a *nominal* belief-state sequence and the optimization is over an action sequence $u_0, u_1, \ldots, u_{H-1}$. The nominal belief-state sequence for ith target can be identified

with the nominal tracks $(\hat{\xi}_k^i, \hat{\mathbf{P}}_k^i)$, which are obtained from the Kalman filter equations [33,34] with exactly zero-noise sequence as follows:

$$\begin{aligned}
\hat{b}_k^{\chi^i}(\chi) &= \mathcal{N}\left(\chi - \hat{\xi}_k^i, \hat{\mathbf{P}}_k^i\right), \qquad (4.3)\\
\hat{\xi}_{k+1}^i &= \mathbf{F}_k \hat{\xi}_k^i,\\
\hat{\mathbf{P}}_{k+1}^i &= \begin{cases} \left[[\hat{\mathbf{P}}_{k+1|k}^i]^{-1} + \mathbf{S}_{k+1}^i\right]^{-1} & \text{if measurement} \\ & \text{available,} \\ \hat{\mathbf{P}}_{k+1|k}^i & \text{otherwise,} \end{cases}
\end{aligned}$$

where

$$\begin{aligned}
\hat{\mathbf{P}}_{k+1|k}^i &= \mathbf{F}_k \hat{\mathbf{P}}_k^i \mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k,\\
\mathbf{S}_{k+1}^i &= \mathbf{H}_{k+1}^{\mathrm{T}} \left[\mathbf{R}_{k+1}\left(\hat{\xi}_{k+1}^i, s_{k+1}\right)\right]^{-1} \mathbf{H}_{k+1}
\end{aligned}$$

and $s_{k+1} = \psi(s_k, u_k)$ (ψ is defined in the next subsection). In equation (4.3), the nominal error covariance matrix i.e., $\hat{\mathbf{P}}_{k+1}^i$ depends on the availability of the observations in the future time. Since the availability of these observations cannot be known for certain, we can guess by assuming the location of the target at time $k+1$ as $\hat{\xi}_{k+1}^{i,\mathrm{pos}}$ (component of nominal track-state corresponding to the ith target at time $k+1$) and checking its line of sight from the sensor location, i.e., s_{k+1}^{pos}. The cost function, i.e., the mean-squared error between the tracks and the targets, can be written as $c(\hat{b}_k, u_k) = \sum_{i=1}^{N_{\mathrm{targs}}} \mathrm{Tr}\, \hat{\mathbf{P}}_{k+1}^i$. Therefore, the goal is to find an action sequence $(u_0, u_1, \ldots, u_{H-1})$ that minimizes the cumulative cost function (truncated horizon [11])

$$J_H(b_0) = \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\mathrm{targs}}} \mathrm{Tr}\, \hat{\mathbf{P}}_{k+1}^i,$$

where $\hat{\mathbf{P}}_{k+1}^i$ represents the nominal error covariance matrix of the ith target at time $k+1$. Here, we adopt an approach called "receding horizon control," according to which we optimize the action sequence for H time steps at the current time-step and implement only the action corresponding to the current time-step and again optimize the action sequence for H time-steps in the next time-step. When there are multiple UAVs, the nominal covariance

matrix for the ith target (based on data fusion techniques) at time $k+1$ is expressed as follows:

$$\hat{\mathbf{P}}^{i}_{k+1} = \left[\sum_{j=1}^{N_{\text{sens}}} \left(\hat{\mathbf{P}}^{i,j}_{k+1}\right)^{-1}\right]^{-1},$$

where N_{sens} represents the number of UAVs and $\hat{\mathbf{P}}^{i,j}_{k+1}$ is the nominal covariance matrix of the the ith target computed at the jth sensor.

Our approach is related to model predictive control (MPC), as argued by the authors of [36]. According to [36], the MPC method is a type of rollout algorithm (an approximation method to solve MDPs and POMDPs) with a particular base policy, where the stability property of MPC is a special case of the cost improvement property of rollout algorithms that employ a sequentially improving base policy. In other words, MPC can also be viewed as an approach to solve a POMDP.

The measurement error, i.e., w_k in (4.1) is distributed according to the normal distribution $\mathcal{N}\left(0, \mathbf{R}_k\left(\chi_k, s_k\right)\right)$, where $\mathbf{R}_k$ reflects $p\%$ range uncertainty and q radian angular uncertainty. If r_k is the distance between the target and the sensor at time k, then the standard deviations corresponding to the range ($\sigma_{\text{range}}(k)$) and the angle ($\sigma_{\text{angle}}(k)$) are written as $\sigma_{\text{range}}(k) = (p/100) * r_k$ and $\sigma_{\text{angle}}(k) = q * r_k$. The information matrix depends on the inverse of the measurement covariance matrix, which depends on the distance between the sensor and the target. Therefore, the information matrix blows up when the UAV is exactly on top of the target (i.e., when $r_k = 0$ the sensor's location overlaps with the target's location in our 2-D environment). To address this problem, we define the effective distance (r_{eff}) between the sensor and the target as follows: $r_{\text{eff}}(k) = \sqrt{r_k^2 + b^2}$, where r_k is the actual distance between the target and the sensor and b is some non-zero real value. Therefore, the standard deviations of the range and the angle are given by $\sigma_{\text{range}}(k) = (p/100) r_{\text{eff}}(k)$ and $\sigma_{\text{angle}}(k) = q r_{\text{eff}}(k)$. If θ_k is the angle between the target and the sensor at time k, then $\mathbf{R}_k$ is calculated as follows:

$$\mathbf{R}_k = M_k \begin{bmatrix} \sigma^2_{\text{range}}(k) & 0 \\ 0 & \sigma^2_{\text{angle}}(k) \end{bmatrix} M_k^{\text{T}}$$

where

$$M_k = \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix}.$$

Clearly, the eigenvalues of the matrix $\mathbf{R}_k$ are $\{\sigma^2_{\text{range}}(k), \sigma^2_{\text{angle}}(k)\}$.

4.3.4 Stationary Target Scenario: Performance Bounds

Let $(u_0^*, u_1^*, \ldots, u_{H-1}^*)$ be the optimal action sequence and J_H^* be the corresponding optimal cumulative cost. Let $(\mathbf{P}_1^{*i}, \mathbf{P}_2^{*i}, \ldots, \mathbf{P}_H^{*i})$ be the sequence of error covariance matrices for the ith target obtained from the optimal action sequence $(u_0^*, \ldots, u_{H-1}^*)$ given a set of prior covariance matrices for all target at time $k = 0$. Therefore, the optimal cumulative cost function is given by

$$J_H^* = \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\text{targs}}} \text{Tr}\, \mathbf{P}_{k+1}^{*i}.$$

We now derive bounds on J_H^* for the stationary target scenario. In general, it is difficult to derive bounds on the optimal cumulative cost for a dynamic-target scenario. So, we chose the stationary target scenario, making the derivation of these bounds tractable. Having these bounds also allows us to compare (albeit indirectly) the performance of the NBO policy with that of the optimal policy (discussed in Subsection 4.3.6.3).

Proposition 3.1: In a stationary-target scenario (i.e., the targets are located at fixed locations and are immobile at all times), if $(\mathbf{P}_0^1, \mathbf{P}_0^2, \ldots, \mathbf{P}_0^{N_{\text{targs}}})$ are the (prior) covariance matrices of targets at time $k = 0$ and are positive definite, then

$$\sum_{i=1}^{N_{\text{targs}}} \sum_{k=0}^{H-1} \frac{M^2}{\text{Tr}\left[(\mathbf{P}_0^i)^{-1}\right] + (k+1)C} \leq J_H^* \leq H \sum_{i=1}^{N_{\text{targs}}} \text{Tr}\, \mathbf{P}_0^i,$$

where M is the rank of the matrix $\mathbf{P}_0^i$, which is the same for all i and

$$C = 2/\min\{(pb/100)^2, (qb)^2\}.$$

Proof: **Upper Bound.** Let $(\mathbf{P}_1^i, \mathbf{P}_2^i, \ldots, \mathbf{P}_H^i)$ be the error covariance matrices for the ith target associated with an arbitrary action sequence $(u_0, \ldots, u_{H-1})$. Let J_H represents the cumulative cost function corresponding to this sequence. Therefore,

$$J_H = \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\text{targs}}} \text{Tr}\, \mathbf{P}_{k+1}^i.$$

Since the targets are stationary, the state evolution for the ith target is given by $\chi^i_{k+1} = \chi^i_k$ $\forall i \in \{1, \ldots, N_{\text{targs}}\}$. Therefore, for any target i and for $k \geq 0$, if the observation is available, then

$$\mathbf{P}^i_{k+1} = \mathbf{P}^i_k - \mathbf{P}^i_k \mathbf{H}^{\mathrm{T}}_k \left[\mathbf{H}_k \mathbf{P}^i_k \mathbf{H}^{\mathrm{T}}_k + \mathbf{R}_k\right]^{-1} \mathbf{P}^i_k, \tag{4.4}$$

and if the observation is not available, then $\mathbf{P}^i_{k+1} = \mathbf{P}^i_k$. When the observation is available,

$$\operatorname{Tr} \mathbf{P}^i_{k+1} = \operatorname{Tr} \mathbf{P}^i_k - \operatorname{Tr} \left[\mathbf{P}^i_k \mathbf{H}^{\mathrm{T}}_k (\mathbf{H}_k \mathbf{P}^i_k \mathbf{H}^{\mathrm{T}}_k + \mathbf{R}_k)^{-1} \mathbf{P}^i_k\right],$$

which implies $\operatorname{Tr} \mathbf{P}^i_{k+1} \leq \operatorname{Tr} \mathbf{P}^i_k$, since the second term in RHS of (4.4), i.e., $\mathbf{P}^i_k \mathbf{H}^{\mathrm{T}}_k (\mathbf{H}_k \mathbf{P}^i_k \mathbf{H}^{\mathrm{T}}_k + \mathbf{R}_k)^{-1} \mathbf{P}^i_k$ is a positive semi-definite matrix, the trace of which is non-negative. Therefore, for both cases, i.e., when the observation is available and also when not available, the following is true: $\operatorname{Tr} \mathbf{P}^i_{k+1} \leq \operatorname{Tr} \mathbf{P}^i_k$ for all $k \geq 0$ and for all i. Therefore, it can be easily verified that for all i and for all $k \geq 0$, $\operatorname{Tr} \mathbf{P}^i_k \leq \operatorname{Tr} \mathbf{P}^i_0$. Therefore,

$$J^*_H \leq J_H \leq H \sum_{i=1}^{N_{\text{targs}}} \operatorname{Tr} \mathbf{P}^i_0.$$

Lower Bound. According to the information filtering (inverse covariance filtering) equations, for any stationary-target i at time $k = 1$,

$$(\mathbf{P}^i_1)^{-1} = (\mathbf{P}^i_0)^{-1} + \mathbf{H}^{\mathrm{T}} \mathbf{R}^{-1}_0 \mathbf{H},$$

which implies

$$\operatorname{Tr} \left[(\mathbf{P}^i_1)^{-1}\right] = \operatorname{Tr} \left[(\mathbf{P}^i_0)^{-1}\right] + \operatorname{Tr} \left[\mathbf{H}^{\mathrm{T}} \mathbf{R}^{-1}_0 \mathbf{H}\right].$$

Since the prior covariance matrices corresponding to each target at $k = 0$ are positive definite, we can easily verify from the information filtering equations that $\mathbf{P}^i_k$ is positive definite for all k and for all i. Since, the observation model is of the form $\mathbf{H} = [\mathbf{I}, \mathbf{0}]$,

$$\operatorname{Tr} \left[(\mathbf{P}^i_1)^{-1}\right] = \operatorname{Tr} \left[(\mathbf{P}^i_0)^{-1}\right] + \operatorname{Tr} \left[\mathbf{R}^{-1}_0\right].$$

For any k, the eigenvalues of $\mathbf{R}_k$ are $\{\sigma^2_{\text{range}}(k), \sigma^2_{\text{angle}}(k)\}$. Therefore,

$$\operatorname{Tr} \left[\mathbf{R}^{-1}_0\right] = 1/\sigma^2_{\text{range}}(0) + 1/\sigma^2_{\text{angle}}(0) \leq C,$$

where $C = 2/\min\{(xb/100)^2, (yb)^2\}$. This inequality can be easily verified from the previous subsection (towards the end), where the description of the measurement covariance matrix is provided.

If $(\lambda_1, \ldots, \lambda_M)$ are the eigenvalues of a positive definite matrix $\mathbf{A}$, then the eigenvalues of $\mathbf{A}^{-1}$ are $(1/\lambda_1, \ldots, 1/\lambda_M)$. From Cauchy's inequality,

$$\sum_{i=1}^{M} \lambda_i \geq M^2 / \sum_{i=1}^{M} 1/\lambda_i.$$

Since the trace of a matrix is the sum of its eigenvalues, $\mathrm{Tr}\,\mathbf{A} \geq M^2/\mathrm{Tr}\,[\mathbf{A}^{-1}]$. From this argument and the previous argument,

$$\mathrm{Tr}\,\mathbf{P}_1^i \geq M^2/\mathrm{Tr}\,\left[(\mathbf{P}_1^i)^{-1}\right] \geq M^2/\left(\mathrm{Tr}\,\left[(\mathbf{P}_0^i)^{-1}\right] + C\right).$$

Similarly, we can extend the above inequality for any time-step k, which is as follows:

$$\mathrm{Tr}\,\mathbf{P}_k^i \geq M^2/\left(\mathrm{Tr}\,\left[(\mathbf{P}_{k-1}^i)^{-1}\right] + C\right) \geq M^2/\left(\mathrm{Tr}\,\left[(\mathbf{P}_0^i)^{-1}\right] + kC\right).$$

Therefore, the cumulative cost function obtained from the action sequence $(u_0, \ldots, u_{H-1})$ satisfies the following inequality:

$$J_H = \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\mathrm{targs}}} \mathrm{Tr}\left(\mathbf{P}_{k+1}^i\right) \geq \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\mathrm{targs}}} M^2/\left(\mathrm{Tr}\,\left[(\mathbf{P}_0^i)^{-1}\right] + C(k+1)\right).$$

The above inequality holds true for every possible action sequence, so it also holds true for the optimal action sequence $(u_0^*, u_1^*, \ldots, u_{H-1}^*)$. Therefore, the following is true:

$$J_H^* \geq \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\mathrm{targs}}} M^2/\left(\mathrm{Tr}\,\left[(\mathbf{P}_0^i)^{-1}\right] + (k+1)C\right).$$

■

4.3.5 UAV Kinematics

In this subsection, we define the mapping function ψ introduced in Section 4.3 to describe the evolution of the sensor (UAV) state given an action, i.e., $s_{k+1} = \psi(s_k, u_k)$. The state of the ith UAV at time k is given by $s_k^i = \left(p_k^i, q_k^i, V_k^i, \theta_k^i\right)$, where (p_k^i, q_k^i) represents the position

coordinates, V_k^i represents the speed, and θ_k^i represents the heading angle. Let a_k^i be the forward acceleration (control variable) and ϕ_k^i be the bank angle (control variable) of the UAV, i.e., $u_k^i = (a_k^i, \phi_k^i)$. The mapping function ψ can be specified as a collection of simple kinematic equations that govern the UAV motion. The kinematic equations of the UAV motion [37] are as follows:

$$
\begin{aligned}
V_{k+1}^i &= \left[V_k^i + a_k^i T\right]_{V_{\min}}^{V_{\max}}, \\
\theta_{k+1}^i &= \theta_k^i + (gT \tan(\phi_k^i)/V_k^i), \\
p_{k+1}^i &= p_k^i + V_k^i T \cos(\theta_k^i), \\
q_{k+1}^i &= q_k^i + V_k^i T \sin(\theta_k^i),
\end{aligned}
$$

where

$$
[v]_{V_{\min}}^{V_{\max}} = \max\left\{V_{\min}, \min(V_{\max}, v)\right\},
$$

where $V_{\min}$ and $V_{\max}$ are the minimum and the maximum limits on the speed of the UAVs.

4.3.6 Empirical Study of NBO

The NBO method is implemented in MATLAB, where the optimization problem discussed in Section 4.3.3 is solved using the command *fmincon* (gradient-based search algorithm). The measurement error, i.e., w_k in (4.1) is distributed according to the normal distribution $\mathcal{N}\left(0, \mathbf{R}_k\left(\chi_k, s_k\right)\right)$, where $\mathbf{R}_k$ reflects 10% range uncertainty and 0.01π radian angular uncertainty. In all our simulations, the time horizon H is set to be $H = 6$. Setting $H = 6$ means that at the current time-step, we plan for 6 time-steps into the future, implement the control action for the current time-step, and discard the next 5 planned actions (these were actions computed purely for planning purposes). At the very next time step, we re-plan for 6 time-steps, and so forth. This kind of planning is called "receding horizon control." The goal of this empirical study is to illustrate the flexibility of the POMDP framework in being able to incorporate features of interest simply by "plugging in" the appropriate feature model into the framework, and to compare the quantitative performance over varying parameter values.

This subsection shows the simulation of a few scenarios to demonstrate the coordinated behavior among the UAVs and the ability of the algorithm to track multiple targets with a single UAV. This subsection also provides empirical results to evaluate how the agility in UAV motion affects the quantitative tracking performance. In these simulations, the trajectory of a target is represented by a sequence of circles and the trajectory of a UAV is represented by a curve joining the arrows (size of an arrow is proportional to the instantaneous speed of the UAV) that point toward the heading direction of the UAV.

4.3.6.1 Coordinated UAV Motion

Figure 4.1 shows the simulation of a scenario with three UAVs and two targets, which depicts the scenario at the end of the simulation. Both targets start at the bottom, and as the simulation progresses, one target moves towards the north-east and the other target moves towards the north-west. In all our simulations, the targets move at a constant speed. The UAVs start at the bottom and move according to the kinematic equations in Section 4.3.5, with controls obtained from the command *fmincon*, which minimize the cost function. The UAVs coordinate to maximize the coverage of the targets as shown in Figure 4.1. There is no explicit assignment of the UAVs to the targets. We are only optimizing the control action vector, which specifies the actions (bank angles and forward accelerations) corresponding to all UAVs. The behavior of the UAVs in Figure 4.1, i.e., the left-most UAV following the left-most target, is an "emergent property" of our approach. To evaluate how the agility in UAV motion affects the quantitative tracking performance, the scenario in Figure 4.1 is simulated for 1000 Monte Carlo runs with variable-speed UAVs and then with fixed-speed UAVs. This allows us to contrast the tracking performance (measured in average target location error) of variable-speed UAVs with that of the fixed-speed UAVs. Figure 4.2 shows the cumulative frequency of the average target-location errors for both fixed-speed and variable-speed UAVs. It is evident from Figure 4.2 that variable-speed UAVs give better performance over fixed-speed UAVs.

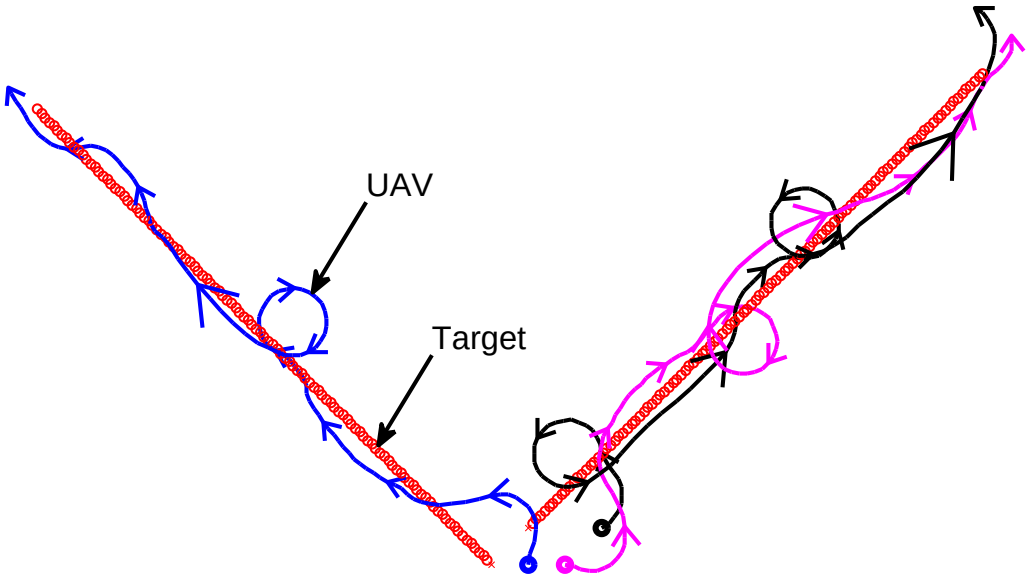


Figure 4.1: Three UAVs tracking two targets.

4.3.6.2 Weaving Between Targets

Figure 4.3 shows the simulation of a scenario with one UAV and two targets, which depicts the scenario at the end of the simulation. Both targets start from the left and move towards the right with constant speed. The UAV weaves between the targets as shown in Figure 4.3, so that the average speed of the UAV towards the right is close to the speed of the targets. Also, the tracks get refined by looking at the targets from different angles, which is achieved by weaving between the targets. This scenario demonstrates the ability of the planning algorithm to maximize the coverage of multiple targets with a single UAV.

4.3.6.3 Stationary Targets

We simulate a scenario with one UAV and one stationary (immobile) target, as shown in Figure 4.4. The stationary target motion model is used to represent the target dynamics, i.e., $\chi_{k+1} = \chi_k$. The bounds in proposition 3.1 are derived for the optimal cumulative cost

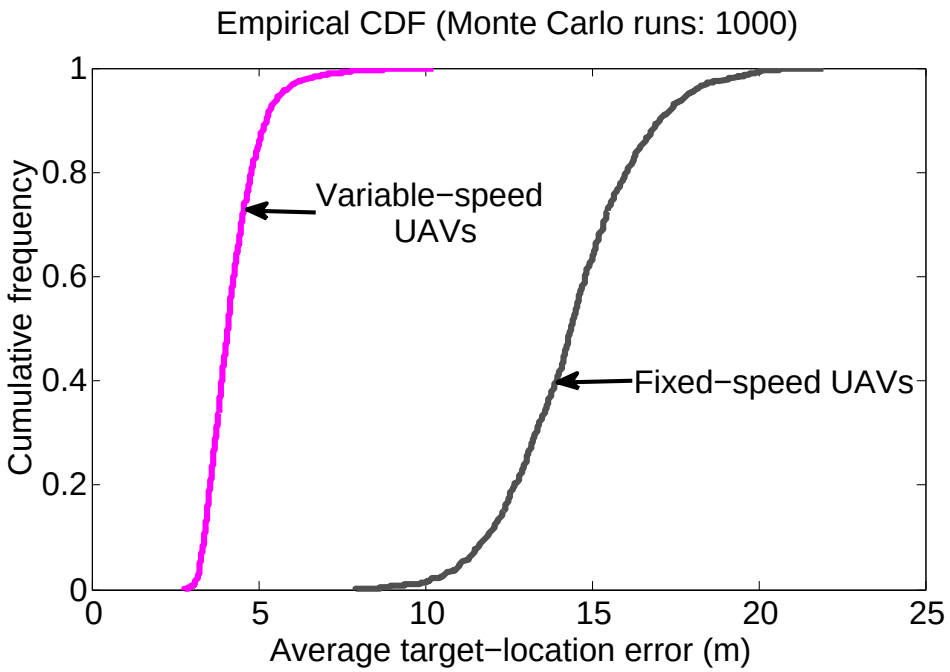


Figure 4.2: Performance comparison: Variable-speed UAVs (speed: 11 m/s - 26 m/s) vs. fixed-speed UAVs (speed: 15 m/s).

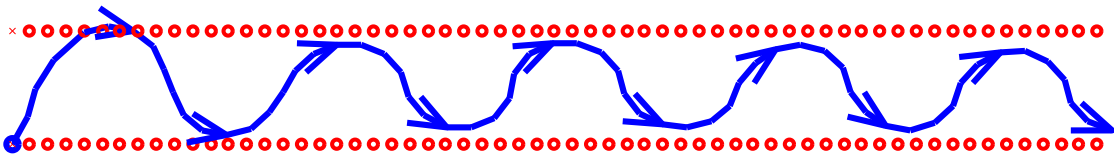


Figure 4.3: One UAV tracking two targets.

starting at time $k = 0$ over horizon H (i.e., till time $k = H - 1$), given prior covariances at time $k = 0$. These bound can be easily extended to optimal cumulative cost starting at any arbitrary time $k > 0$ over horizon H (i.e., till time $k + H - 1$) by replacing the prior covariances $\mathbf{P}_0^i$ in proposition 3.1 with $\mathbf{P}_k^i$. For the scenario in Figure 4.4, we plot (as shown in Figure 4.5) the cumulative cost over horizon $H = 6$ obtained from the NBO policy ($J_H^{\text{NBO}}(k)$) as a function of time k along with the lower and upper bounds on the optimal

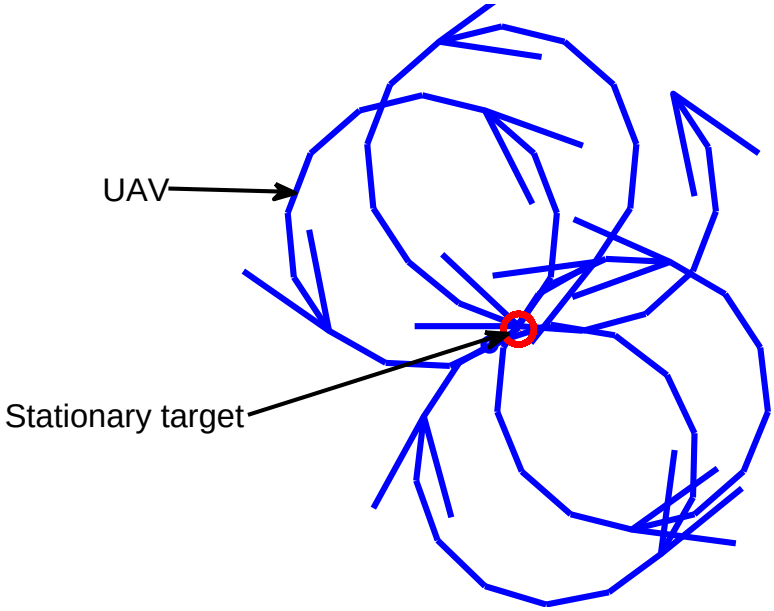


Figure 4.4: UAV tracking a stationary target.

cumulative cost given by

$$J_H^{\mathrm{LB}}(k) = \sum_{i=1}^{N_{\mathrm{targs}}} \sum_{j=k}^{k+H-1} M^2/\mathrm{Tr}\left[(\mathbf{P}_k^i)^{-1}\right] + C(j+1),$$

$$J_H^{\mathrm{UB}}(k) = H \sum_{i=1}^{N_{\mathrm{targs}}} \mathrm{Tr}\,\mathbf{P}_k^i$$

respectively. Figure 4.5 shows that the cumulative cost of the NBO policy is close to the lower bound of the optimal cumulative cost, which suggests that NBO is close to optimal.

4.4 Wind Compensation

If unaccounted for, wind drags the UAVs from their planned paths, which results in tracking performance degradation. This section presents a wind-compensation method in the context of the POMDP framework. More precisely, we incorporate the state of wind (i.e., wind speed in x and y directions) into the framework to nullify its effect on UAVs. To model the dynamics of wind, we use autoregressive model of order p. This model has been used in the past by several researchers (e.g., [38, 39]) to model the dynamics of the wind.

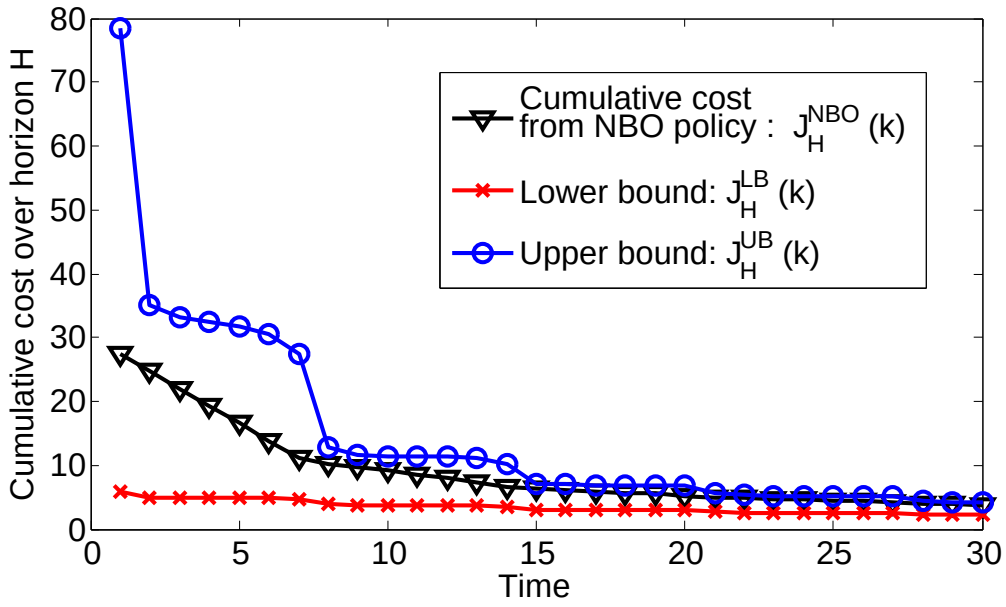


Figure 4.5: Cumulative cost from the NBO policy for the scenario in Figure 4.4 along with its lower and upper bounds.

Let w_k^x and w_k^y represent the speed of wind in x and y directions respectively at time k, which evolve according to the autoregressive model given by the following equations:

$$w_{k+1}^x = \sum_{i=1}^{p} A_i w_{k-i+1}^x + e_{k+1},$$
$$w_{k+1}^y = \sum_{i=1}^{p} A_i w_{k-i+1}^y + e_{k+1},$$

where $(A_1, \ldots, A_p)$ represents the model coefficients and $e_{k+1} \sim \mathcal{N}(0, \mathbf{Q}_{\text{wind}})$. Here, we assume that the wind speeds in x and y directions vary only with time and not space. This wind model is incorporated into the POMDP framework by including the state of the wind into the POMDP state space. Here, we set $p = 3$. We define an augmented wind-speed vector (for $x-$direction) as follows: $W_k^x = [w_k^x, w_{k-1}^x, w_{k-2}^x]^{\mathrm{T}}$. The wind-speed vector evolves according to

$$W_k^x = \mathbf{A}_{\text{autoreg}} W_{k-1}^x + E_k, \tag{4.5}$$

where

$$\mathbf{A}_{\text{autoreg}} = \begin{bmatrix} A_1 & A_2 & A_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

and $E_k = [e_k, 0, 0]$, where $e_k \sim \mathcal{N}(0, \mathbf{Q}_{\text{wind}})$. The evolution of the wind-speed vector for y-direction can be written similarly as above.

The NBO method requires a nominal belief-state sequence of future states of the targets and the sensors (UAVs) given the actions. Since our planning algorithm has a *lookahead* property, it is important to have a UAV motion model that correctly predicts the trajectory of each UAV. Figure 4.6(a) shows the simulation of a scenario with one UAV and one target in the presence of wind. In the simulations, the wind speed in the north direction is dominant compared to the wind speed in the east direction (this kind of behavior can be set at the start of the simulation by appropriately initializing the wind-speeds in x and y directions). Figure 4.6(a) shows a snapshot at the end of the simulation where wind was unaccounted for, which resulted in drifting of the UAV from its desired course. As a result, the tracking performance (measured in average target-location error) gets deteriorated.

To nullify the effect of wind, the estimated wind speeds are incorporated into the kinematics of the UAV motion. The modified location update equations are as follows:

$$p^i_{k+1} = p^i_k + V^i_k T \cos(\theta^i_k) + w^x_{k,\text{est}} T,$$
$$q^i_{k+1} = q^i_k + V^i_k T \sin(\theta^i_k) + w^y_{k,\text{est}} T,$$

where $w^x_{k,\text{est}}$ and $w^y_{k,\text{est}}$ are the estimates of wind speeds in x and y directions at time k. The speed and bank angle update equations remain the same as in Subsection 4.3.5. The wind speed estimates are obtained by tracking the wind-speed vector using the Kalman filter. In general, aircrafts measure the true airspeeds (speed of aircraft with respect to the surrounding atmosphere) via a device called pitot tube. Since the aircraft knows its actual speed and direction (through GPS, Radio NAV etc.), the wind speed and direction can be computed. This justifies the availability of the measurements of the wind state at each time

epoch. The observation models for wind speeds in x and y directions are given by

$$W^x_{k,obs} = \mathbf{H}_{\text{wind}} W^x_k + n^x_k,$$
$$W^y_{k,obs} = \mathbf{H}_{\text{wind}} W^y_k + n^y_k,$$

where $\mathbf{H}_{\text{wind}} = [1, 0, 0]$, $n^x_k \sim \mathcal{N}(0, \mathbf{R}^x_{\text{wind}})$ and $n^y_k \sim \mathcal{N}(0, \mathbf{R}^y_{\text{wind}})$. Therefore, the estimates of wind, i.e., $w^x_{k,\text{est}}$ and $w^y_{k,\text{est}}$ are obtained from the Kalman filter equations given the state evolution model (4.5) and the observation model (as above). Therefore,

$$w^x_{k,\text{est}} = W^x_{k,est}[1],$$
$$w^y_{k,\text{est}} = W^y_{k,est}[1],$$

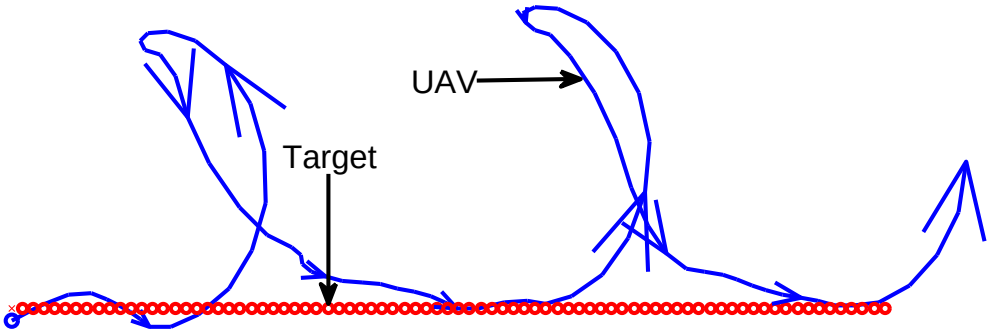
where $W^x_{k,est}[1]$ represents the first element of the Kalman filter estimate of the wind-speed vector (in x-direction) $W^x_{k,est}$ and $W^y_{k,est}[1]$ represents the first element of the Kalman filter estimate of the wind-speed vector (in y-direction) $W^y_{k,est}$. The nominal estimates of wind-speed vectors over the time horizon H (required to compute the approximate objective function for NBO method) is obtained by evolving the Kalman filter equations for wind-state with exactly zero noise sequence as follows:

$$\hat{W}^x_{k+1,est} = \mathbf{A}_{\text{autoreg}} \hat{W}^x_{k,est},$$
$$\hat{W}^y_{k+1,est} = \mathbf{A}_{\text{autoreg}} \hat{W}^y_{k,est},$$

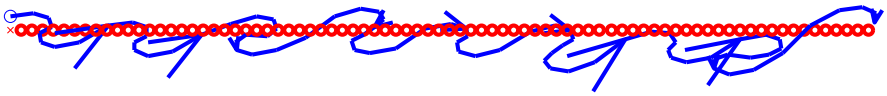
where $\hat{W}^x_{k,est}$ and $\hat{W}^y_{k,est}$ are the nominal estimates of the wind-speed vectors at time k for x and y directions respectively. This wind-compensation method is not limited to just autoregressive wind-model (as in our example); we can incorporate any wind model into our framework. A more rigorous method for estimating wind parameters can be found in [40].

Figure 4.6(b) shows the simulation of the same scenario as in Figure 4.6(a), but with modified UAV kinematic model (as discussed before). With wind compensation, the UAV trajectory stays close to that of the target's, which improves the tracking performance. This scenario is simulated (with and without wind compensation) for 500 runs and the average

target-location error is calculated at the end of every run, which is plotted in Figure 4.7. From Figure 4.7, it is evident that the wind-compensation method improves the tracking performance significantly compared to the case where wind was unaccounted for.



(a) No wind compensation.



(b) Wind compensation.

Figure 4.6: UAV tracking a target in the presence of wind.

4.5 Collision Avoidance

4.5.1 Collision avoidance between UAVs and obstacles

Suppose that the targets are moving on the ground in the presence of obstacles where the UAV flying altitude is lower than the height of the obstacles. A collision between the UAVs and the obstacles may happen if the obstacles are unaccounted for. These obstacles also act as occlusions, i.e., a sensor cannot generate the measurement of a target if it is occluded. The authors in [11] did treat occlusions but not collision avoidance. This section shows how to avoid collisions between UAVs and obstacles in the context of POMDP framework. To avoid

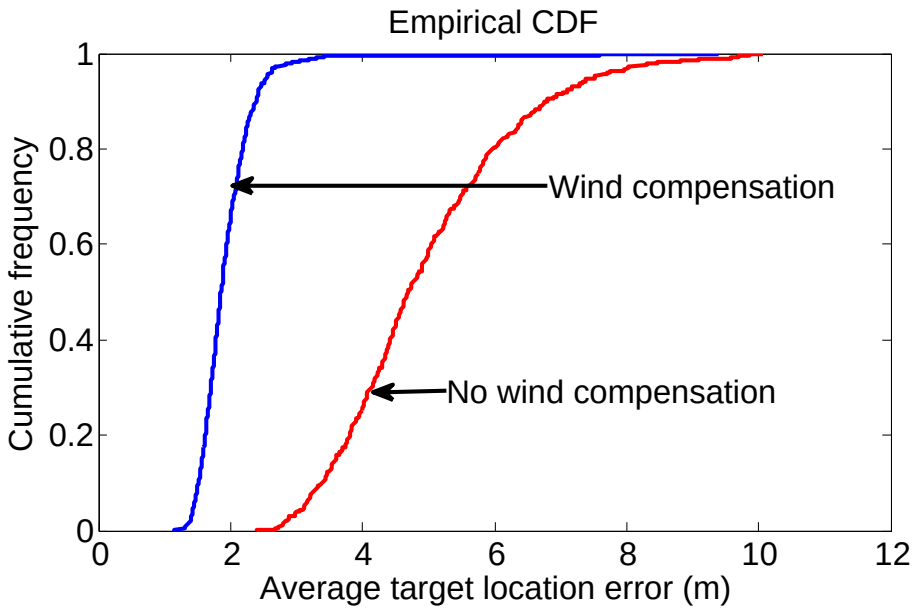


Figure 4.7: Tracking performance comparison.

collisions, we include a penalty term (that increases as the UAVs come close to obstacles) in the objective function. Let N_{targs} represent the number of targets and N_{sens} represent the number of UAVs. The new objective function is written as follows:

$$J_H = \sum_{k=0}^{H-1} \left(\sum_{i=1}^{N_{\text{targs}}} \text{Tr}\, \hat{\mathbf{P}}_{k+1}^{i} + \gamma \sum_{j=1}^{N_{\text{sens}}} P_{k+1}^{\text{coll},j} \right), \tag{4.6}$$

where $P_{k+1}^{\text{coll},j}$ is a collision penalty function corresponding to the jth UAV and γ is a scaling factor. For the purpose of our 2-D simulation environment, we model the obstacles as regions of various shapes in the plane (surveillance region), i.e., square, rectangle, and circle. We can use any continuous function as the collision penalty function provided that the function increases as a UAV approaches an obstacle. For our simulation, we use a simple linear function as the penalty function as follows:

$$P_{k+1}^{\text{coll},j} = D - d_{k+1}^{j}$$

if $d_{k+1}^{j} < D$, and $P_{k+1}^{\text{coll},j} = 0$ otherwise, where d_{k+1}^{j} is the distance between the jth UAV and the closest obstacle from its location at time $k+1$ and D is a constant such that the

penalty function is non-zero only when d^j_{k+1} is less than D (the idea is that the obstacles should not affect the planning algorithm when the UAVs are far from the obstacles). We assume that a UAV can detect its distance from the surrounding obstacles. Therefore, d^j_{k+1} $(\forall j)$ is available at all times. Since our controller acts at discrete time epochs, at each time epoch (or decision epoch), only one obstacle effects our decision. However, if a UAV is in between two obstacles, then, the closest obstacle could alternate between the two obstacles from one time epoch to the next, leading to a situation called "chattering." In the scope of our work here, we do not concern ourselves with the practical undesirable ramifications of this kind of rapid switching or chattering. Indeed, in practice some filtering of the command signal might be necessary to smooth out the command over time to mitigate the possible "chattering."

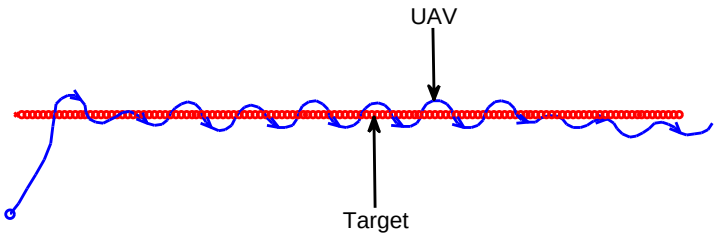
A scenario is simulated with one UAV and one target, where the target moves in the presence of obstacles. Figure 4.8(b) shows a snapshot at the end of the simulation. Both the target and the UAV start at the left and as the target moves towards the right, the UAV follows the target while evading the obstacles. If there were no obstacles, the UAV trajectory would have been close to that of the target as in Figure 4.8(a). When the UAV goes far from the target (when avoiding obstacles), the trace objective increases. However, the penalty from the collision penalty function is reduced (γ in (4.6) is set to a sufficiently large value to induce this behavior).

4.5.2 Collision avoidance among UAVs

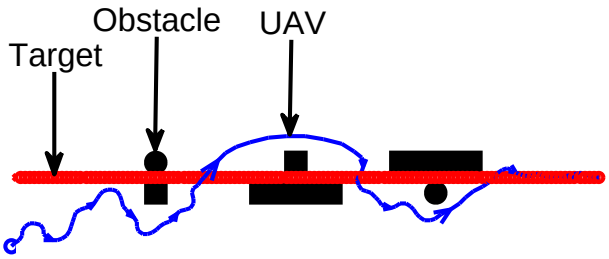
Consider a multi-UAV scenario, where all UAVs are flying at the same altitude. To avoid collisions among the UAVs, a penalty function is incorporated into the objective function similar to (4.6), as follows:

$$J_H = \sum_{k=0}^{H-1} \left(\sum_{i=1}^{N_{\text{targs}}} \operatorname{Tr} \hat{\mathbf{P}}^i_{k+1} + \gamma \sum_{j=1}^{N_{\text{sens}}} P^{\text{coll},j}_{k+1} \right),$$

where $P^{\text{coll},j}_{k+1} = D - d^j_{k+1}$ if $d^j_{k+1} < D$, and $P^{\text{coll},j}_{k+1} = 0$ otherwise, where d^j_{k+1} is the distance between the jth UAV and the closest neighboring UAV at time $k+1$, i.e., $d^j_{k+1} = \min_{i,i\neq j} d^{ji}_{k+1}$,



(a) No obstacles.



(b) Obstacles in the path of the UAV.

Figure 4.8: UAV tracking a target while evading obstacles.

where d_{k+1}^{ji} is the distance between the jth UAV and the ith UAV. The significance of D (above) is that a UAV contributes to the penalty function only if it approaches another UAV such that the distance between them is less than D. This penalty function penalizes those actions (over time) that cause any two UAVs to come close to each other, which effectively avoids collisions among the UAVs. Suppose that the safe-distance (minimum spacing between two aircrafts, which is maintained to avoid mid-air collisions) between two aircrafts is 100m. The value of D is set equal to the safe-distance, i.e., $D = 100$. Figure 4.9 shows the simulation of a scenario where two UAVs are tracking one target with enhancement to the objective function. To demonstrate the effectiveness of our collision avoidance enhancement, we plot the distance between the two UAVs as a function of time for the scenario in Figure 4.9 for $\gamma = 10$. From Figure 4.10, it is evident that, with the enhancement to the objective function (as discussed above), the distance between the UAVs is greater than D,

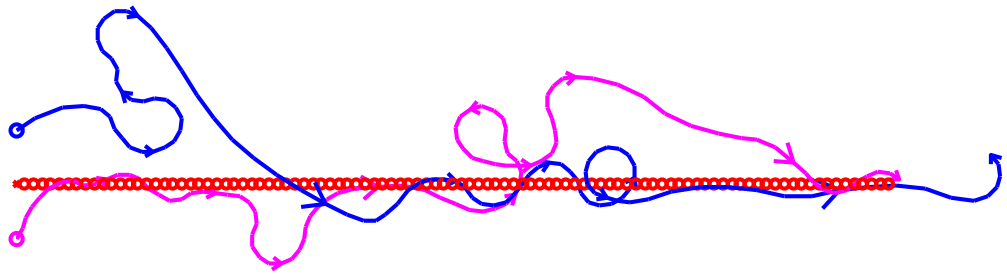


Figure 4.9: Two UAVs tracking a target while avoiding collisions.

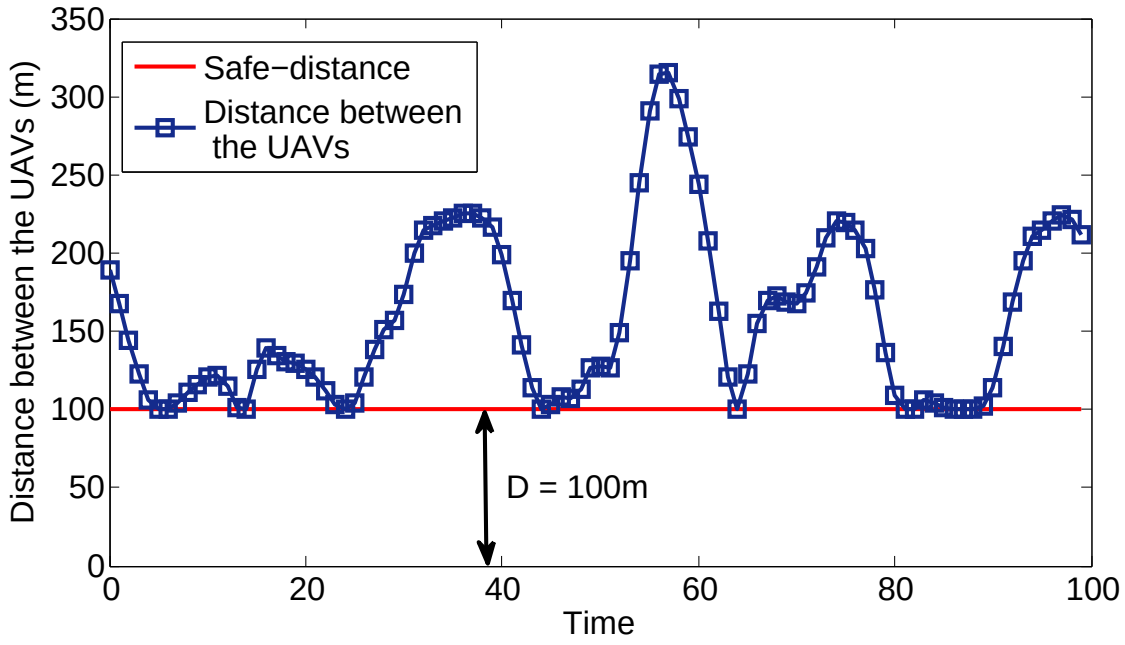


Figure 4.10: Distance between the UAVs as a function of time (for scenario in Figure 4.9).

i.e., the safe-distance, at all times. This shows that the enhancement to the objective function enables the planning algorithm to guide the UAVs for tracking a target while avoiding collisions among the UAVs by keeping a safe distance (D) between them.

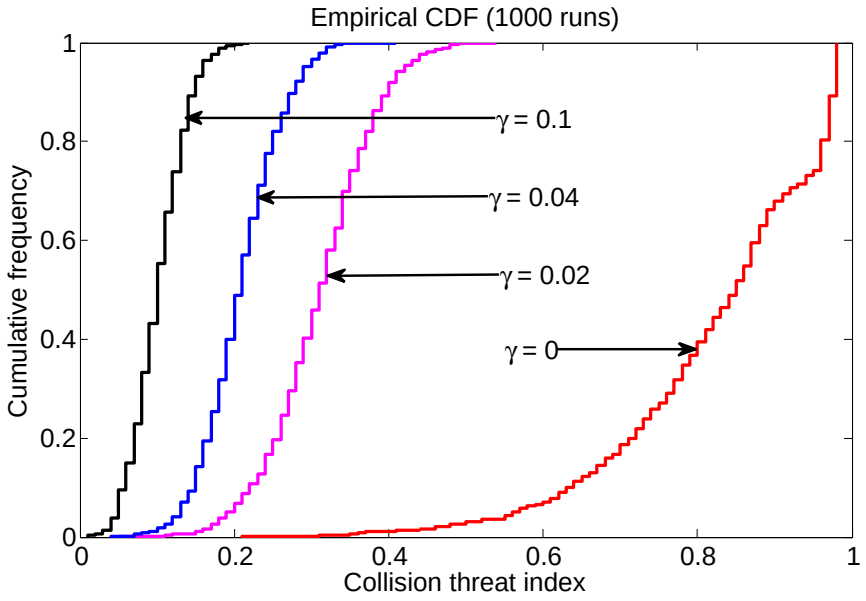
We introduce a performance metric called *Collision Threat Index* (CTI) for the scenario in Figure 4.9, which is defined as the fraction of time the distance between the UAVs is less than D. The scenario in Figure 4.9 is simulated for 1000 Monte Carlo runs for various values of γ in (4.6). Figure 4.11(a) shows the plot of the cumulative frequency of CTI for various values of γ. It is evident from Figure 4.11(a) that the performance with respect to CTI can be improved by increasing the value of γ. However, increasing the value of γ reduces the tracking performance with respect to the average target location error, which is evident from Figure 4.11(b). Figure 4.11(b) shows the plot of the cumulative frequency of average target location errors for various values of γ. In summary, γ can be used to tune the tradeoff between the performance with respect to CTI and the performance with respect to average target location error. This approach can be easily extended to 3-D collision avoidance maneuver by: 1) defining a 3-D motion model for UAVs; and 2) having a penalty metric (similar to the one we had for the 2-D case) to penalize the control actions that bring two UAVs close to each other.

4.6 Evading Threats

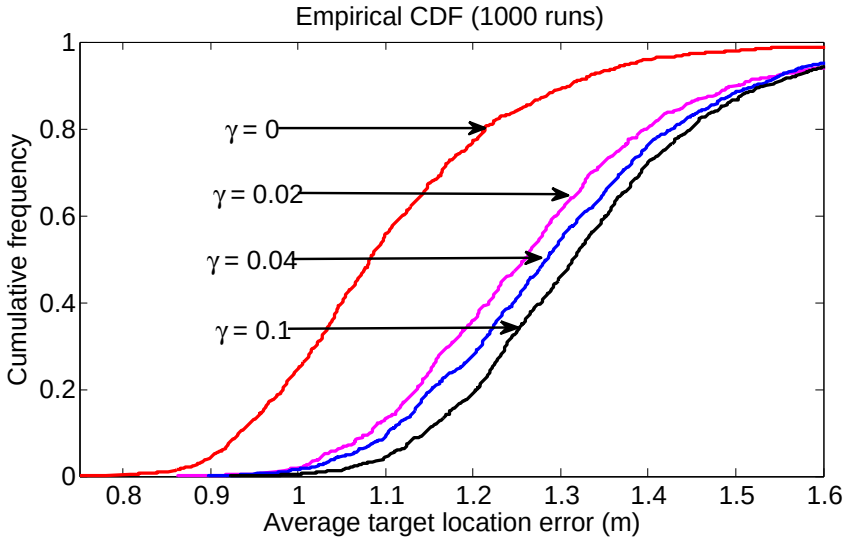
For the purpose of this section, a *threat* is a ground vehicle that actively pursues UAVs that are tracking it. Each threat knows the locations of all UAVs at all times and aligns its direction of motion toward the closest UAV at every time-step. Our goal is to design the guidance algorithm in such a way that the UAVs track targets while avoiding threats by keeping a safe distance from them. The next subsection defines the threat motion model that describes the evolution of the threat state.

4.6.1 Threat Motion Model

Let χ_k^t represent the state of a threat at time k, which includes the position coordinates (x_k and y_k) and speed (v_k) of the threat, i.e., $\chi_k^t = [x_k, y_k, v_k]^{\mathrm{T}}$. The threat state evolves



(a) CTI.



(b) Average target location error.

Figure 4.11: Cumulative frequency of performance measures for various values of γ in (4.6).

according to the following equation:

$$\chi_{k+1}^t = \mathbf{\Phi}_k \chi_k^t + n_k,\ n_k \sim \mathcal{N}(0, \mathbf{Q}_k^t), \tag{4.7}$$

where $\mathbf{Q}_k^t$ represents the process noise covariance matrix and

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & 0 & \cos(h_k)T \\ 0 & 1 & \sin(h_k)T \\ 0 & 0 & 1 \end{bmatrix},$$

where h_k is the heading angle of the threat, which is given by

$$h_k = \tan^{-1}\left(\frac{y_k^{\text{closest UAV}} - y_k}{x_k^{\text{closest UAV}} - x_k}\right),$$

where $(x_k^{\text{closest UAV}},\ y_k^{\text{closest UAV}})$ represent the position coordinates of the closest UAV from the threat at time k and $(x_k,\ y_k)$ represent the position coordinates of the threat at time k. The heading angle equation (above) captures the pursuing property of the threat.

4.6.2 Threat belief state evolution

The observation of the threat state is given by $z_k^t = \mathbf{H}_k\chi_k^t + w_k$, $w_k \sim \mathcal{N}(0, \mathbf{R}_k)$. Since the threat state evolves according to (4.7), i.e., $\chi_{k+1}^t = f(\chi_k^t) + n_k$, the Kalman filter or the extended Kalman filter cannot be used to evaluate the threat state estimate because the function $f(\cdot)$ is nonlinear and not differentiable. Therefore, a heuristic approach is adopted to write the Kalman filter equations to evaluate the threat state estimate. The threat belief state can be expressed (or approximated) as: $b_k^t(\chi) = \mathcal{N}(\chi - \xi_k^t, \mathbf{P}_k^t)$, where ξ_k^t and $\mathbf{P}_k^t$ evolve according to a heuristic Kalman filter (differs from the Kalman filter only in the prediction step), where the prediction step is written as follows: $\xi_{k|k-1}^t = \hat{\mathbf{\Phi}}_k\xi_{k-1}^t$ and $\mathbf{P}_{k|k-1}^t = \hat{\mathbf{\Phi}}_k\mathbf{P}_{k-1}^t\hat{\mathbf{\Phi}}_k^{\mathrm{T}} + \mathbf{Q}_k^t$, where $\hat{\mathbf{\Phi}}_k$ is the approximation of $\mathbf{\Phi}_k$ in (4.7), which is evaluated as follows:

$$\hat{\mathbf{\Phi}}_k = \begin{bmatrix} 1 & 0 & \cos(\hat{h}_k)T \\ 0 & 1 & \sin(\hat{h}_k)T \\ 0 & 0 & 1 \end{bmatrix},$$

where

$$\hat{h}_k = \tan^{-1}\left(\frac{\hat{y}_k^{\text{closest UAV}} - \hat{y}_k}{\hat{x}_k^{\text{closest UAV}} - \hat{x}_k}\right), \tag{4.8}$$

where ($\hat{x}_k^{\text{closest UAV}}$, $\hat{y}_k^{\text{closest UAV}}$) are the position coordinates of the estimated closest UAV from the threat at time k and ($\hat{x}_k$, $\hat{y}_k$) are the estimated position coordinates of the threat at time k. The closest UAV from the threat can be found by evaluating the Euclidean distances between the threat and each UAV. The location of the threat is known only with uncertainty, which is given by the posterior distribution of the threat state. Therefore, we cannot evaluate the exact Euclidean distance between the threat and a UAV. So, we evaluate the Mahalanobis distance [41], which is a statistical distance, between the threat and a UAV. The closest UAV from the threat is estimated as follows: 1) We evaluate the Mahalanobis distance [41] of the position coordinates of each UAV i ($s_k^{i,\text{pos}}$) from the probability distribution (Gaussian) of the threat's location given by $\mathcal{N}(\xi_k^{t,\text{pos}}, \mathbf{P}_k^{t,\text{pos}})$, where $\xi_k^{t,\text{pos}}$ represents the first two elements of the threat state estimate ξ_k^t and $\mathbf{P}_k^{t,\text{pos}}$ represents a 2×2 sub-matrix of the threat state error covariance matrix $\mathbf{P}_k^t$, such that

$$\mathbf{P}_k^t = \begin{bmatrix} \mathbf{P}_k^{t,\text{pos}} & \mathbf{A}_k \\ \mathbf{A}_k^{\text{T}} & \mathbf{B}_k \end{bmatrix}.$$

The Mahalanobis distance of the ith UAV from the probability distribution of the threat's location is given by

$$D_{\text{M}}^i = \sqrt{[s_k^{i,\text{pos}} - \xi_k^{t,\text{pos}}]^{\text{T}} [\mathbf{P}_k^{t,\text{pos}}]^{-1} [s_k^{i,\text{pos}} - \xi_k^{t,\text{pos}}]}.$$

2) We estimate the closest UAV I from the threat as follows: $I = \arg\min_i D_{\text{M}}^i$.

The *nominal* threat belief state ($\hat{b}_k^{\chi,t}$) is identified with the *nominal* tracks of the threat ($\hat{\xi}_k^t$,$\hat{\mathbf{P}}_k^t$) as follows:

$$\begin{aligned} \hat{b}_k^{\chi,t}(\chi) &= \mathcal{N}\left(\chi - \hat{\xi}_k^t, \hat{\mathbf{P}}_k^t\right), \\ \hat{\xi}_{k+1}^t &= \hat{\mathbf{\Phi}}_k \hat{\xi}_k^t, \\ \hat{\mathbf{P}}_{k+1}^t &= \left[(\hat{\mathbf{P}}_{k+1|k}^t)^{-1} + \mathbf{S}_{k+1}\right]^{-1}, \end{aligned}$$

where

$$\hat{\mathbf{P}}^t_{k+1|k} = \hat{\mathbf{\Phi}}_k \hat{\mathbf{P}}^t_k \hat{\mathbf{\Phi}}^{\mathrm{T}}_k + \mathbf{Q}^t_k,$$
$$\mathbf{S}_{k+1} = \mathbf{H}^{\mathrm{T}}_{k+1} \left[\mathbf{R}_{k+1} \left(\hat{\xi}^t_{k+1}, s_{k+1} \right) \right]^{-1} \mathbf{H}_{k+1}.$$

Similar to the previous sections, we incorporate the *constant velocity* (CV) model (4.2) for target dynamics into the NBO method. The next subsection describes an enhancement to the objective function to guide UAVs for tracking targets while evading threats.

4.6.3 Enhancement to the objective function

Let N_{targs} represent the number of targets and N_{sens} represent the number of UAVs. To guide the UAVs for tracking targets while avoiding threats, we include a penalty metric in the objective function. This penalty term increases whenever a UAV comes close to a threat. Therefore the Euclidean distance between the locations of a UAV and the closest threat from the UAV can be used as a penalty metric. As mentioned in the previous section, we cannot evaluate the Euclidean distance between a UAV and a threat because of the uncertainty in the location of the threat. Therefore, as before, we use the Mahalanobis distance [41], which is a statistical distance, between a UAV and its closest threat as the penalty metric. The new objective function is written as follows:

$$J_H(b_0) = \sum_{k=0}^{H-1} \left(\sum_{i=1}^{N_{\mathrm{targs}}} \mathrm{Tr}\, \hat{\mathbf{P}}^i_{k+1} + \gamma \sum_{j=1}^{N_{\mathrm{sens}}} G^j_{k+1} \right), \tag{4.9}$$

where γ is a scaling constant and

$$G^j_{k+1} = \begin{cases} D - d^j_{k+1} & \text{if} \quad d^j_{k+1} < D, \\ 0 & \text{otherwise,} \end{cases} \tag{4.10}$$

where $d^j_{k+1} = \min_m D^m_{\mathrm{M}}(s^{j,\mathrm{pos}}_{k+1})$, where $s^{j,\mathrm{pos}}_{k+1}$ represents the position coordinates of the jth UAV and $D^m_{\mathrm{M}}(s^{j,\mathrm{pos}}_{k+1})$ represents the Mahalanobis distance of $s^{j,\mathrm{pos}}_{k+1}$ from the probability distribution of the location of mth threat, i.e., $\mathcal{N}(\hat{\xi}^{t,\mathrm{pos},m}_{k+1}, \hat{\mathbf{P}}^{t,\mathrm{pos},m}_{k+1})$, where $\hat{\xi}^{t,\mathrm{pos},m}_{k+1}$ represents the first two elements of $\hat{\xi}^{t,m}_{k+1}$ and $\hat{\mathbf{P}}^{t,\mathrm{pos},m}_{k+1}$ represents the top-left-corner 2×2 sub-matrix of $\hat{\mathbf{P}}^{t,m}_{k+1}$.

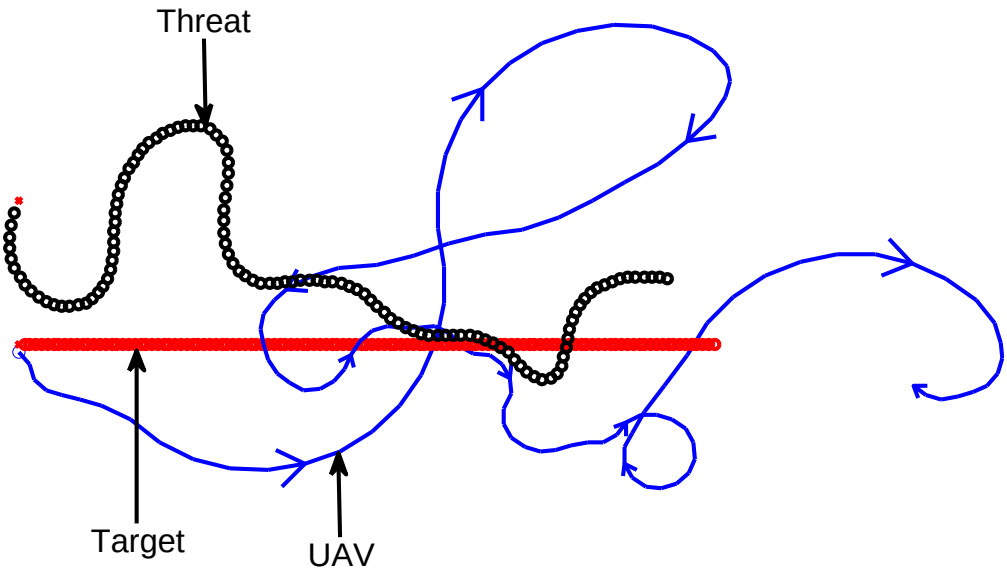


Figure 4.12: UAV tracking a target while evading a threat.

The next subsection demonstrates the effectiveness of this enhancement with an empirical study.

4.6.4 Empirical Study

Figure 4.12 shows the simulation of a scenario with one UAV, one threat, and one target. In this simulation, we set $D = 100$ in (4.10). To demonstrate the effectiveness of the enhancement to the objective function (4.9), we plot the distance $D_{\text{threat-UAV}}$ between the UAV and the threat as a function of time in the scenario Figure 4.12, which is shown in Figure 4.13. It is evident from Figure 4.13 that the UAV tracks the target while maintaining a safe distance, D_{safe} $(= D)$, from the threat.

We introduce a performance metric called the *Threat Index* for the scenario in Figure 4.12, defined as the fraction of time the distance between the UAV and the threat is less than D. We simulate the scenario in Figure 4.12 for 1000 Monte Carlo runs for various values of γ used in (4.9). Figure 4.14(a) shows the plot of the cumulative frequency of threat indices

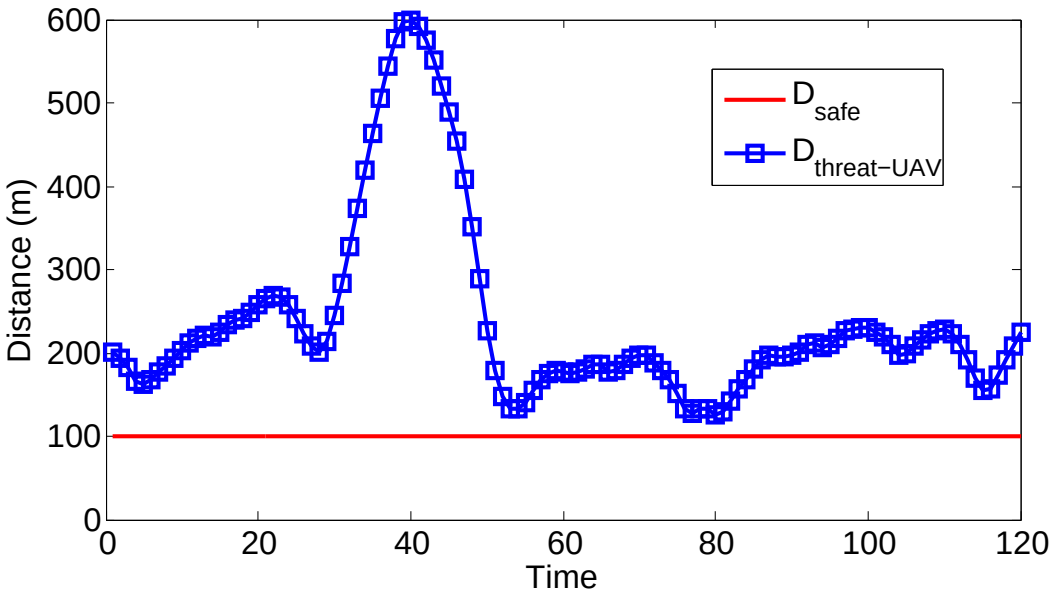
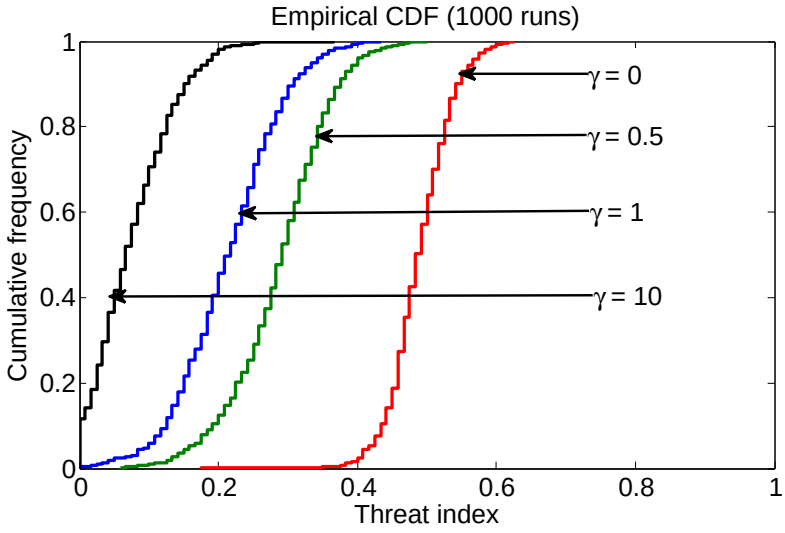


Figure 4.13: Distance between the UAV and the threat as a function of time in the scenario Figure 4.12.

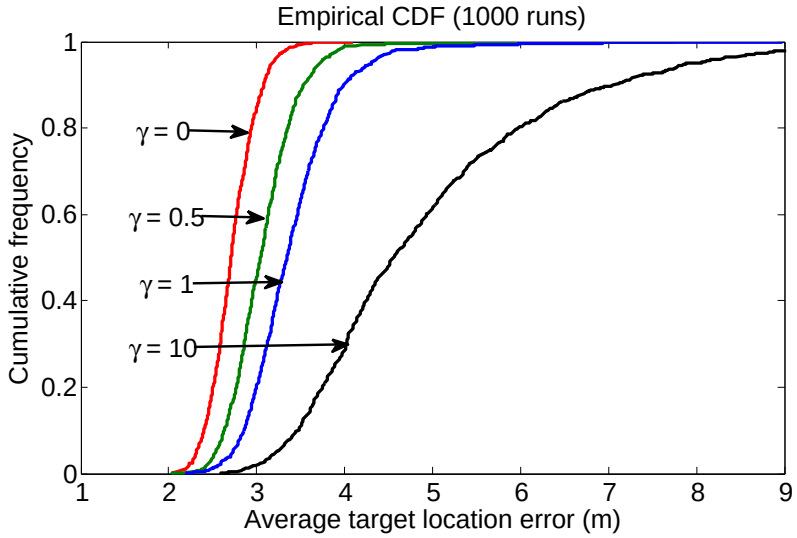
for various values of γ. From Figure 4.14(a), it is evident that the performance with respect to *Threat Index* can be improved by increasing the value of γ. However, the performance with respect to average target location error degrades with increasing γ, which is evident from Figure 4.14(b). In summary, the parameter γ can be used to tune the tradeoff between the performance with respect to *Threat Index* and the performance with respect to average target location error.

4.7 Tracking Evasive Targets

For the purpose of this section, an evasive target is a ground vehicle that actively moves away from UAVs that are tracking it, to avoid being tracked. In this section, all targets are evasive. Each target knows the locations of all UAVs at every time-step. The state of a target evolves according to the evasive target motion model as described in the following subsection.



(a) Threat index.



(b) Average target location error.

Figure 4.14: Cumulative frequency of performance measures for various values of γ in the scenario Figure 4.12.

4.7.1 Evasive Target Motion Model

Our evasive target motion model is very similar to the threat motion model presented in the previous section. Let χ_k represent the state of a target at time k, which includes the position coordinates (x_k and y_k) and speed (v_k), i.e., $\chi_k = [x_k, y_k, v_k]^{\mathrm{T}}$. The target state

evolves according to $\chi_{k+1} = \mathbf{\Phi}_k \chi_k + n_k$, $n_k \sim \mathcal{N}(0, \mathbf{Q}_k^{\text{Evas}})$, where $\mathbf{Q}_k^{\text{Evas}}$ represents the process noise covariance matrix and

$$\mathbf{\Phi}_k = \begin{bmatrix} 1 & 0 & \cos(h_k)T \\ 0 & 1 & \sin(h_k)T \\ 0 & 0 & 1 \end{bmatrix},$$

where h_k is the heading angle of the target, which is given by

$$h_k = \pi + \tan^{-1}\left(\frac{y_k^{\text{closest UAV}} - y_k}{x_k^{\text{closest UAV}} - x_k}\right),$$

where $(x_k^{\text{closest UAV}},\ y_k^{\text{closest UAV}})$ are the position coordinates of the closest UAV from the target at time k and $(x_k,\ y_k)$ are the position coordinates of the target at time k. The above heading angle equation differs from (4.8) in that the former induces an addition of π radians, representing motion away from the closest UAV.

4.7.2 Target Belief State Evolution

As before, the observation of the target state is given by $z_k = \mathbf{H}_k \chi_k + w_k$, $w_k \sim \mathcal{N}(0, \mathbf{R}_k)$. The target belief state b_k^{χ} and the *nominal* target belief state $\hat{b}_k^{\chi}$ evolve according to the heuristic Kalman filter equations as in Section 4.6.2. However, the heading angle estimate (used in $\hat{\mathbf{\Phi}}_k$) is evaluated according to the following equation:

$$\hat{h}_k = \pi + \tan^{-1}\left(\frac{\hat{y}_k^{\text{closest UAV}} - \hat{y}_k}{\hat{x}_k^{\text{closest UAV}} - \hat{x}_k}\right),$$

where $(\hat{x}_k^{\text{closest UAV}},\ \hat{y}_k^{\text{closest UAV}})$ are the position coordinates of the estimated closest UAV from the target at time k (the closest UAV from the target is estimated in the same way we estimated the closest UAV from a threat in the previous section) and $(\hat{x}_k,\ \hat{y}_k)$ are the estimated position coordinates of the target at time k. We incorporate the nominal target belief-state update equations (described above) into the NBO method. The objective function is given by

$$J_H(b_0) = \sum_{k=0}^{H-1}\left(\sum_{i=1}^{N_{\text{targs}}} \text{Tr}\,\hat{\mathbf{P}}_{k+1}^i\right),$$

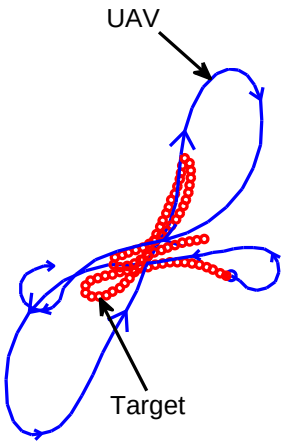
where $\hat{\mathbf{P}}^i_{k+1}$ is the nominal error covariance matrix corresponding to the ith target at time $k+1$, which is obtained from the nominal target belief-state update equations (described above).

4.7.3 Empirical Study

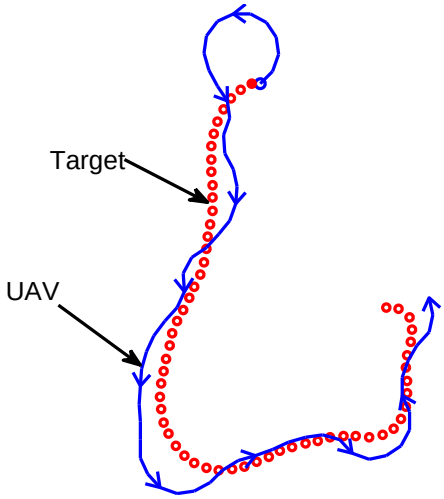
To demonstrate the effectiveness of incorporating the evasive target motion model into the NBO method (as described in Section 4.7.2), we compare its performance with the use of a target motion model that does not account for the evasive property of the target (in particular, *constant velocity* model). First, we simulate a scenario with one UAV and an evasive target while adopting the *constant velocity* model (henceforth referred to as the CV-model) [33,34] for evasive target dynamics (which means the target is evasive, but we are not accounting for the evasive property of the target) (4.2). Second, we simulate the previous scenario, while adopting the evasive target motion model (presented in Section 4.7.2) for target dynamics. The plots of one UAV tracking an evasive target with the CV-model and with the evasive target motion model (Evas-model) are shown in Figure 4.15(a) and Figure 4.15(b) respectively. We simulate these scenarios for 1000 Monte Carlo runs and plot the cumulative frequency of average target location errors, as shown in Figure 4.16. From Figure 4.16, it is evident that by adopting the Evas-model over the CV-model for the dynamics of an evasive target, the performance with respect to average target location error is improved significantly.

4.8 Track Swap Avoidance

A track swap is a switch in the association between the tracks and the targets. The identities of the targets are interpreted through the association between the tracks and the targets. Therefore, a track swap switches the identities of targets, which is undesirable. The likelihood of a track swap depends on the tracker state, which in-turn depends on the locations of UAVs relative to the targets. Therefore, the track swaps can be mitigated by appropriately controlling the UAVs. This is achieved by incorporating a metric into the



(a) NBO with CV-model.



(b) NBO with Evas-model.

Figure 4.15: UAV tracking an evasive target.

objective function that represents the risk of a track swap. Although this enhancement was introduced in [11], our work builds on it by incorporating multiple candidate metrics and comparing their performance for variable-speed UAVs via Monte Carlo simulations.

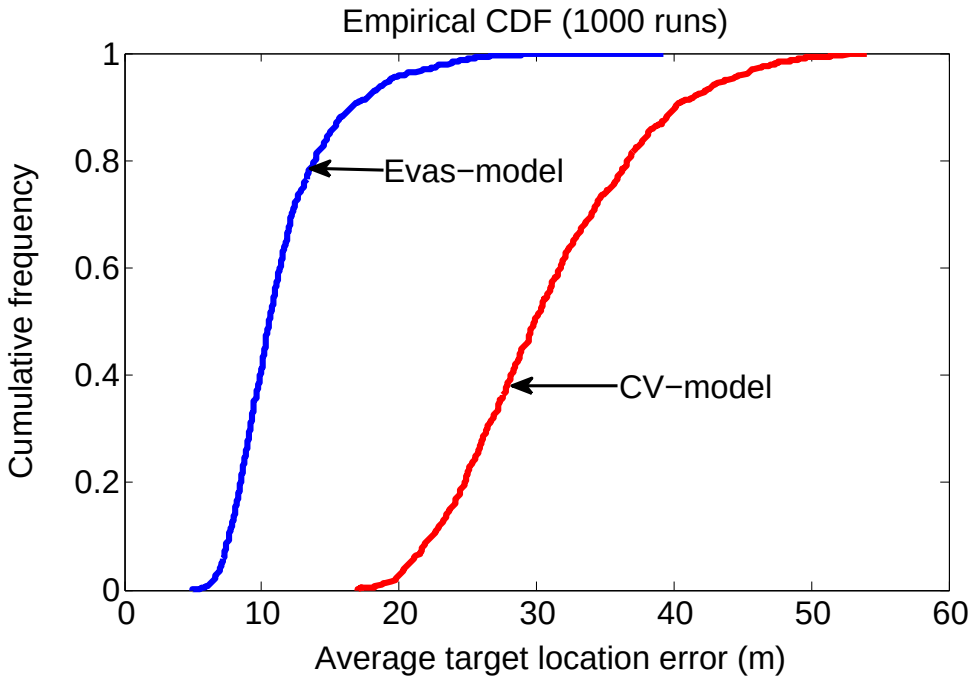


Figure 4.16: Performance comparison of Evas-model and CV-model for the scenarios in Figure 4.15.

4.8.1 Problem Description

Suppose that a UAV is tracking three targets as shown in Figure 4.17. In this scenario, the bottom two targets come in close proximity to each other periodically and the topmost target remains far from the bottom two targets. Our planning algorithm maximizes the coverage of the targets by guiding the UAV to weave between the topmost target and the bottom targets (as shown in Figure 4.17), which is achieved by minimizing the overall trace objective. The likelihood of a track swap is high when the measurement sources are ambiguous. In the scenario of Figure 4.17, when the UAV is far from the bottom targets, i.e., when the UAV is close to the topmost target, the likelihood of a track swap (corresponding to the bottom targets) is high because the chance that the sources of the measurements from the bottom two targets becoming ambiguous is high. The following subsection presents an enhancement [11] to mitigate track swaps.

4.8.2 Enhancement for Mitigating Track Swaps

The similarity between the target state distributions is a good predictor for a track swap because the likelihood of the measurement sources becoming ambiguous is high when the similarity between the target state distributions is high. The target state distributions depend on the tracker states and the locations of the UAVs over time. Therefore, we can control the UAVs appropriately such that the target state distributions are less similar. The similarity between the probability distributions can be measured as the inverse of a statistical distance between the distributions. To minimize the likelihood of a track swap, we incorporate a term that is inversely proportional to a statistical distance (between the target state distributions) into the objective function. The new objective function is written as

$$J_H = \sum_k \left(\sum_{i=1}^{N_{\text{targs}}} \operatorname{Tr} \mathbf{P}_{k+1}^i + \gamma(1/D_{k+1}) \right),$$

where γ is a scaling factor, N_{targs} represents the number of targets, and

$$D_{k+1} = \min_{p \neq q;\, p,q \in \mathcal{S}} D(\chi_{k+1}^p || \chi_{k+1}^q),$$

where $\mathcal{S} = \{1, 2, \ldots, N_{\text{targs}}\}$ and $D(\chi_{k+1}^p || \chi_{k+1}^q)$ is a statistical distance between the distributions of the pth target and the qth target.

There are several statistical distances defined in the literature: KL-divergence [42], Bhattacharya distance [43], Hellinger distance [44], and worst-case chi-square distance (see [11] for a description of the worst-case chi-square distance). We simulate a scenario with three targets and one UAV with the above new objective function (where we use the worst-case chi-square distance to measure the similarity between distributions) as shown in Figure 4.18, where the UAV stays close to the bottom targets in contrast to the behavior in Figure 4.17. This reduces the similarity between the distributions of the bottom targets, which reduces the chance of a track swap. KL-divergence, Bhattacharya distance, and Hellinger distance are average-case measures of how often the state values from the distributions fall in a small neighborhood of each other, whereas worst-case chi-square distance is a worst-case measure.

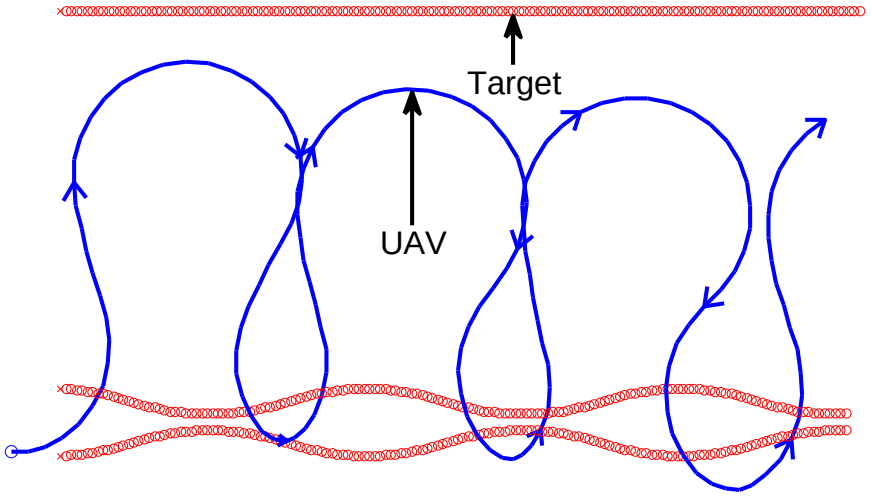


Figure 4.17: UAV tracking three targets.

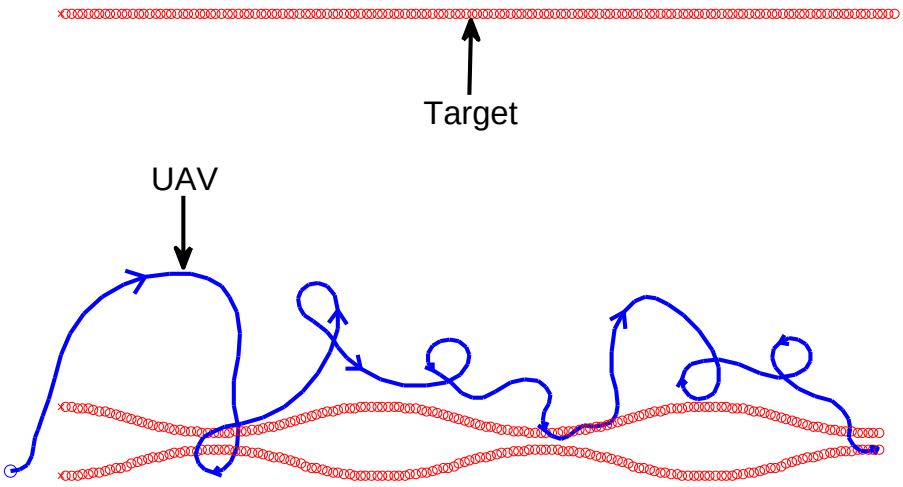


Figure 4.18: UAV tracking three targets while mitigating track swaps.

The authors of [11] explain that a worst-case measure is more suited to track swap prediction than an average-case measure, which is corroborated in the next subsection via an empirical study.

4.8.3 Empirical Study

We simulate 2000 runs of the scenario where a UAV tracks three targets (as described in Section 4.8.1) with and without the enhancement in the objective function. In every run, we calculate the fraction of track swaps—a metric to evaluate the tracking performance—according to the following method.

Fraction of Track Swaps: Let N_{TS} represent the number of track swaps, defined as follows. We check for a track swap only at instances when the bottom targets are the farthest apart. These instances occur periodically as can be seen in Figure 4.17 and Figure 4.18. At each of these instances, we evaluate the associations between the tracks and the targets. Whenever the association at a particular instance differs from the association in the previous instance, we increment N_{TS} by one. At the end of the simulation, we evaluate the fraction of track swaps as follows: fraction of track swaps $= \frac{N_{\text{TS}}}{N}$, where N represents the total number of instances we evaluate the track associations.

In our simulations, we use the following notation to represent the statistical distances: 1) KL-divergence: $D_{\text{KL-div}}$, 2) Hellinger distance: $D_{\text{Hell-dist}}$, 3) Bhattacharya distance: $D_{\text{Bhatt-dist}}$, and 4) worst-case chi-square distance: D_{χ^2}. Figure 4.19 shows the plot of the cumulative frequency of fraction of track swaps for various statistical distances. From Figure 4.19, it is evident that the enhancement to the objective function improved the performance, with respect to the fraction of track swaps, significantly for all candidate statistical distances. It is also evident that D_{χ^2} is the best among other statistical distances in mitigating track swaps. This shows that the worst-case chi-square distance is an appropriate metric for predicting track swaps—corroborating the claim in [11].

4.9 Concluding Remarks

The results from Subsection 4.3.6 show that variable-speed UAVs give better tracking performance over fixed-speed UAVs, which demonstrates that the higher the agility in UAV motion, the better the tracking performance. Agile UAVs give better performance because

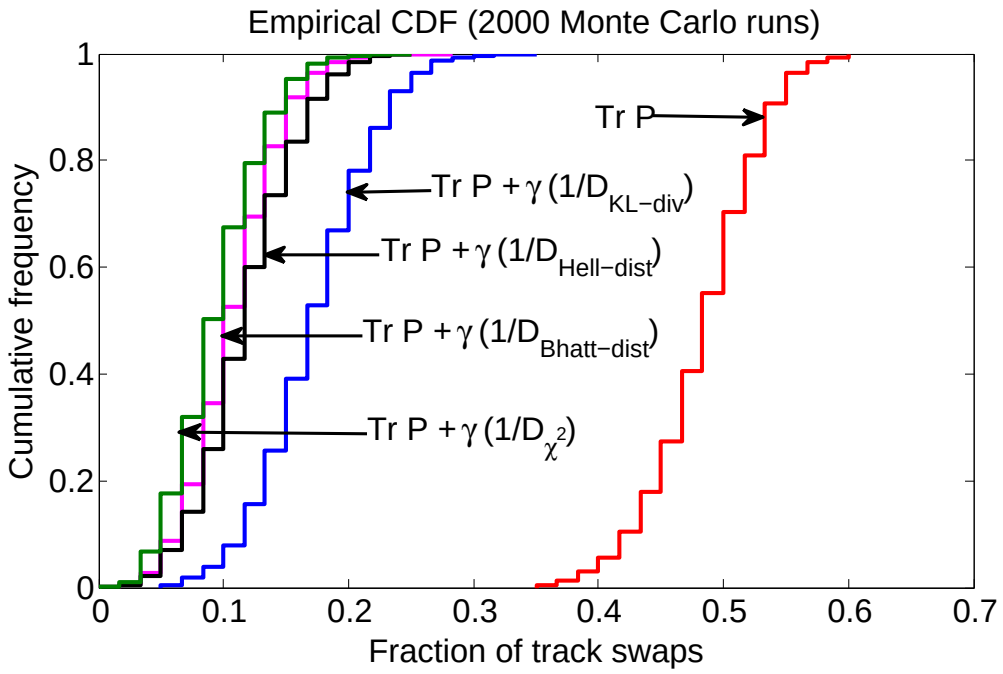


Figure 4.19: Performance comparison for various statistical distances.

they can quickly change the geometry compared to fixed-speed UAVs. The results from Subsection 4.3.6.3 show that the cumulative cost of the NBO policy is close to the lower bound of the optimal cumulative cost, which suggests that NBO is close to optimal. The results from Section 4.6 demonstrate that the target location error is strongly influenced by the UAV information enhancement maneuvers, which in turn depend on the threat motion model. The UAVs try to move close to the targets to increase the tracking performance, and at the same time they try to move away from threats in order to increase the performance with respect to *threat index*. If the UAVs try to maximize only the tracking performance, then the performance with respect to *threat index* degrades, and if the UAVs try to maximize only the performance with respect to *threat index*, then the tracking performance degrades, i.e., the goals of maximizing these two performances conflict each other. This implies that we need to have a control on the tradeoff between these two performances. Therefore, we show in this chapter (in Section 4.6) that the tradeoff between the performances with respect

to *threat index* and average target location error can be tuned with a parameter γ. The authors of [11] explain that a worst-case measure is more suited to track swap prediction than the following average-case measures: KL-divergence, Bhattacharya distance, and Hellinger distance. An average-case measure tells how often the state values from the distributions fall in a small neighborhood of each other, whereas worst-case chi-square distance [11] is a worst-case measure. The empirical results from Section 4.8 support the argument that the worst-case chi-square distance D_{χ^2} is an appropriate metric in predicting track swaps over other statistical distances. The results in this chapter were published in [45, 46].

CHAPTER 5

DECENTRALIZED GUIDANCE CONTROL OF UAVS WITH EXPLICIT OPTIMIZATION OF COMMUNICATION

5.1 Introduction

There has been a growing interest in the development of guidance methods for unmanned aerial vehicles (UAVs) for target tracking. Our work in the past [11, 45, 46] was focused on centralized guidance, where there was a notional central controller that collects measurements from the sensors on-board the UAVs, forms tracks on the targets, computes control commands for the UAVs, and sends them back to the UAVs. But that approach did not account for the cost of communication. In general, the communication among the UAVs or between UAVs and a central controller (e.g., ground station) involves a nonzero cost.

The decentralized control of UAVs was studied before in various contexts, e.g., decentralized UAV formation flight [47, 48], decentralized cooperative search in an uncertain environment by UAVs [49], decentralized model predictive control of UAVs [50], and decentralized task assignment for UAVs [51]. In this study, we design a decentralized guidance control method for UAVs for target tracking based on the theory of *decentralized partially observable Markov decision process* (Dec-POMDP). Each UAV has a fixed communication range, i.e., two UAVs can directly (with no relays) communicate with each other only if the distance between them is less than a constant value. The communication network among the UAVs is dynamic in the sense that a communication link between two UAVs forms when they are within each other's communication range, and breaks otherwise. At each time instance, any two UAVs can communicate with each other if there exists a path between them through

the network. The communication among the UAVs involves a nonzero cost, and we explicitly optimize the communication decisions (at the network level) of each UAV (with whom to communicate and when to communicate) along with the kinematic controls. In this approach, a UAV collects measurements of targets from the on-board sensors, forms tracks on the targets based on the local observations and the information received from other UAVs, and computes the joint kinematic control commands and the communication decisions. This kind of communication-aware motion planning for mobile agents has been studied before, e.g., [52]. However, our approach differs from the existing communication-aware planning approaches in that we place our planning method in the context of Dec-POMDP.

Dec-POMDP [10] is a mathematical framework useful for modeling resource control problems where the decision making is decentralized. The Dec-POMDP approach has the following advantages: 1) it offers a general approach to dealing with resource management in a multi-agent scenario with decentralized control, 2) it is a non-myopic approach, which trades off short-term for long-term performance, 3) in comparison to the existing greedy/myopic approaches in the literature, the Dec-POMDP approach results in more effective exploitation of limited resources in a decentralized setting. In addition, we explicitly optimize the communication between the UAVs (also called agents) at the network level. In general, solving a Dec-POMDP exactly is intractable. Instead, we extend a centralized POMDP (*partially observable Markov decision process*) approximation method to solve the Dec-POMDP. We seek a solution to our guidance control problem that is implementable in real-time (a typical requirement). Therefore, we need an approximation method with computational requirements that are not prohibitive. In this study, we choose an approximation method called *nominal belief-state optimization* (NBO), which we used in the past [11, 45, 46] to solve a centralized POMDP. The NBO method is less expensive (in time-consumption) than other approximation methods.

The Dec-POMDP evolves in discrete time steps, where the length of each time step is T seconds. We use k as the discrete-time index. For the purpose of this study, we assume that the decision epochs at each agent are synchronized.

5.2 System and Problem Description

Targets. The targets are ground-based vehicles and are assumed to be moving in 2-D (our framework easily extends to 3-D, with a concomitant additional computational cost).

Unmanned Aerial Vehicles. For the purpose of this study, we assume that the UAVs fly at a constant altitude, i.e., the UAVs move in 2-D (again, our framework easily extends to 3-D). The kinematics of each UAV is controlled by forward acceleration and bank angle. Each UAV is equipped with an on-board sensor that generates the position measurements of the targets. For simplicity, we assume that the locations and velocities of every UAV in the system are available at each UAV (e.g., by communicating GPS coordinates).

Measurement Error. The sensors on-board each UAV generate measurements of the target locations. The measurement of a target location at a sensor is corrupted by a spatially varying random error that depends on the relative location of the target with respect to the location of the sensor/UAV.

Communication. Each UAV has communication capabilities, i.e., each UAV can transmit and receive information to/from other UAVs, subject to certain constraints. The next section provides a detailed description of the communication between the agents.

Tracker. Each UAV is equipped with a tracker (tracking algorithm) that maintains tracks on each target and updates them via Kalman filter equations given the local observations and the information received from other UAVs.

Objective. Our goal is to control each UAV, in a decentralized setting, such that a cost metric, which includes the mean-squared error between the tracks and the targets and the cost of communication, is minimized (discussed later).

5.3 Communication Between Agents

In this study, the UAVs communicate with each other over a network. Each UAV has a fixed communication range of d_c, i.e., each UAV can transmit or receive information to or from another UAV only if the distance between them is less than d_c. We define the notion of a communication link between a pair of UAVs as follows. At each time step, there exists a communication link between a pair of UAVs if the distance between them is less than d_c, and there exists no link otherwise. A pair of UAVs, not within the communication range of each other, can still exchange information by relaying the information through the network. The network is dynamic, i.e., the links between the UAVs form and break with time. At each time step, a UAV can send or receive information to or from another UAV if there exists a *path* (a sequence of links connecting UAVs) between them through the network. We assume that communication delays are sufficiently small relative to the time duration between decision epochs that any information communicated at discrete-time k is received in time for decision-making at discrete-time $k+1$. To account for the cost of communicating information over the network, we assign a numerical value to each communication link, and we refer to this value as the *cost* of the link. For simplicity, the value of the cost of the link between a pair of agents, if it exists, is assumed to be proportional to the distance between the two associated agents. More precisely, if d_k^{ij} is the distance between the ith and the jth UAVs at time k such that there exists a link between them (i.e., $d_k^{ij} < d_c$), then the cost of the link is αd_k^{ij}, where α is a given proportionality constant. The cost of a path through the network is the sum of the costs of the links that form the path. At any time step, a pair of UAVs cannot communicate with each other if there exists no path between them through the network.

The communication among the agents is restricted in the following ways:

- At each decision epoch, an agent can communicate with at most one other agent. The rationale for this restriction is that allowing for multiple agent destinations per

decision epoch results in prohibitive computational requirements when making optimal communication decisions over a long time horizon (planning over a long horizon is a characteristic of the Dec-POMDP formulation).

- If an agent i decides to communicate with agent j at time k, then the agent i can choose and send to j one of the L locally generated target observations in the past L time steps (including the current-step). We set the value of L to a sufficiently small value so that the computational requirement is not prohibitive.

5.4 Problem Formulation

In this section, we cast the UAV guidance problem into the framework of a *decentralized partially observable Markov decision process* (Dec-POMDP for short). To cast the UAV guidance problem into the Dec-POMDP framework, we need to define the following key components in terms of our guidance problem as follows.

5.4.1 Dec-POMDP Ingredients

Agents. There are N agents (or UAVs) in the system. Let $\mathcal{I} = \{1, \ldots, N\}$ represent the set of agents.

States. We account for the following three subsystems in specifying the state: the UAV(s), the target(s), and the tracker (includes the state of the tracking algorithm at each agent). More precisely, the state at time k is given by $x_k = (s_k, \chi_k, T_k)$, where s_k represents the UAV state, χ_k represents the target state, and T_k represents the joint track state. The UAV state s_k includes the locations and velocities of each UAV. The target state χ_k includes the location and velocity of each target. The joint track state is given by $T_k = (T_k^1, \ldots, T_k^N)$, where $T_k^i = (\xi_k^i, \mathbf{P}_k^i)$ is the state of the tracking algorithm at agent i, where ξ_k^i is the posterior mean vector and $\mathbf{P}_k^i$ is the posterior covariance matrix.

Joint Actions. The joint action is a tuple, the components of which are actions corresponding to individual agents. Let $u_k = (u_k^1, \ldots, u_k^N)$ represent the joint action, where u_k^i is

the action vector at agent i. The local action vector u_k^i includes the kinematic controls and the communication decisions for the ith UAV at time k. The kinematic controls of a UAV includes its forward acceleration and bank angle. The communication decision at agent i at time k can be represented by (g_k^i, l_k^i), where $g_k^i \in \mathcal{I} \cup \{0\} \setminus \{i\}$ is the ID of the agent with whom agent i will communicate at time k ($g_k^i = 0$ means that agent i will not communicate with any other agent at time k), and $l_k^i \in \{k - L, \ldots, k\}$ is the chosen time step from the past L time steps (starting from k) such that the agent i will send to agent g_k^i its local target observation generated at time step l_k^i. The local action at agent i for $i = 1, \ldots, N$ can be represented by $u_k^i = (a_k^i, g_k^i, l_k^i)$, where a_k^i includes the kinematic controls (forward acceleration and bank angle) for agent/UAV i, and (g_k^i, l_k^i) represents its communication decision at time k.

State Transition Law. The state transition law specifies the next-state probability density given the current state and joint action, i.e., $x_{k+1} \sim p_k(\cdot|x_k, u_k)$, where p_k is a conditional probability density. Since we have several sub-systems to describe the state, we define the state-transition law separately for each sub-system. The state of UAV i for $i = 1, \ldots, N$ evolves according to the kinematic equations given the actions/control commands, i.e., $s_{k+1}^i = \psi(s_k^i, a_k^i)$ (see below for an explicit definition of ψ), where a_k^i represents the local kinematic controls (forward acceleration and bank angle). The state of the ith UAV at time k is given by $s_k^i = (p_k^i, q_k^i, V_k^i, \theta_k^i)$, where (p_k^i, q_k^i) represents the position coordinates, V_k^i represents the speed, and θ_k^i represents the heading angle. The kinematic control action for UAV i is given by $a_k^i = (f_k^i, \phi_k^i)$, where f_k^i is the forward acceleration and ϕ_k^i is the bank angle of the UAV. The kinematic equations of the UAV motion [45] are as follows:

$$
\begin{aligned}
V_{k+1}^i &= \left[V_k^i + f_k^i T\right]_{V_{\min}}^{V_{\max}} \\
\theta_{k+1}^i &= \theta_k^i + (gT \tan(\phi_k^i)/V_k^i), \\
p_{k+1}^i &= p_k^i + V_k^i T \cos(\theta_k^i), \\
q_{k+1}^i &= q_k^i + V_k^i T \sin(\theta_k^i),
\end{aligned}
$$

where $[v]_{V_{\min}}^{V_{\max}} = \max\{V_{\min}, \min(V_{\max}, v)\}$, $V_{\min}$ and $V_{\max}$ are the minimum and the maximum limits on the speed of the UAVs, g is the acceleration due to gravity, and T is the length of the time step.

The target state evolves according to

$$\chi_{k+1} = \mathbf{F}\chi_k + e_k,\ e_k \sim \mathcal{N}(0, \mathbf{Q}). \tag{5.1}$$

We use the constant velocity model to represent the kinematics of a target (see [34] or [33] for the definition of $\mathbf{F}$ and $\mathbf{Q}$). Finally, the track state at each agent evolves according to the Kalman filter update equations given the local observations and the observations received from other agents.

Observations and Observation Law. Each joint observation is a tuple, the components of which are observations made by individual agents. Let $z_k = (z_k^1, \ldots, z_k^N)$ be the joint observation vector at time k, where z_k^i represents the observation at agent i. The observation law specifies the probability density of joint observations given the current state and joint action, i.e., $z_k \sim q_k(\cdot|x_k, u_k)$, where q_k is a conditional probability density. The observations at agent i, represented by z_k^i, includes the observation of the UAV state, i.e., the current location and velocity of each UAV, the local track state T_k^i, and the target state. We assume that the UAV and the local track states are fully observable at an agent, and the target state is not fully observable; instead, at each time step the agent has access only to a random measurement of the target state that is a function of the locations of the targets and the agent. Specifically, the observation corresponding to the target state, at agent i for $i = 1, \ldots, N$, is given by

$$z_k^{i,\chi} = \mathbf{H}\chi_k + w_k^i,\ w_k^i \sim \mathcal{N}\left(0, \mathbf{R}\left(\chi_k, s_k^i\right)\right), \tag{5.2}$$

where $\mathbf{H}$ is the observation model and $\mathbf{R}(\cdot)$ is the measurement covariance matrix that depends on the locations of the targets and the agent. We assume that the agents generate noisy observations of the 2-D target positions. An agent i for $i = 1, \ldots, N$ upon receiving information about the target-state (i.e., target observations) from other agents, fuses the

locally generated observations with the observations received from other agents, and updates the local track state.

Cost Function. A cost function $C(x_k, u_k)$ specifies the cost (real number) of being in a given state x_k and performing a joint action u_k. The cost function includes the mean-squared tracking error and the cost of communication (details are discussed later).

The Dec-POMDP starts at a random initial state x_0 (whose probability density is given), and at any typical time step k, the state x_k transitions to x_{k+1} given the joint action vector u_k. The joint action u_k performed at the current state x_k incurs a global cost $C(x_k, u_k)$. As a Dec-POMDP evolves over time as a dynamical process, the agents may not know the underlying state exactly, but each agent generates observations of the underlying state, providing the agent with clues of the actual underlying states. Given the Dec-POMDP formulation, the goal is to find joint actions over a horizon H such that the expected cumulative cost, over a time horizon H, is minimized.

An agent may not know the action taken and the observation generated at another agent. An agent may decide to communicate with another agent, and these decisions to communicate are embedded into the joint action vector u_k. The communication among the agents incurs a cost (as discussed in Section 5.3), which is embedded in the global cost function $C(x_k, u_k)$. The local observations allow each agent to infer, with some uncertainty, what states actually occurred. This uncertainty is represented by the *local belief-state*, which is the *a posteriori* density of the underlying state given the history of local observations and local actions made by that agent, including the information gathered from other agents. Just as in centralized POMDPs, in the decentralized case the local belief-state will be used as "feedback" information that is needed for controlling the system. In other words, we seek an optimal joint policy that depends only on the local belief-states.

5.4.2 Objective and Optimal Policy

The problem is to minimize the cumulative cost over horizon H, given by

$$\mathrm{E}\left[\sum_{k=0}^{H-1} C(x_k, u_k)\right].$$

In the centralized case (if it were a POMDP problem), this objective function can be written in terms of "global" belief-states as follows:

$$J(b_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k, u_k) \middle| b_0\right], \tag{5.3}$$

where $c(b,u) = \int C(x,u)b(x)\,dx$, b_k is the "global" belief-state, i.e., the posterior density at time k, and $\mathrm{E}[\cdot|b_0]$ represents conditional expectation given the initial belief-state b_0 at time $k=0$. The goal is to pick the joint actions over time so that the objective function is minimized. In general, the actions chosen for agents at each time should be allowed to depend on the entire history of observations and actions up to that time. However, if an optimal choice of such a sequence of actions exists, then there is an optimal choice of actions that depends only on "belief-state feedback." Hence, ignoring for the time being the decentralized nature of the problem, what we seek is an optimal *policy*, which maps the belief-state at each time to the joint action tuple at that time. The optimal policy is characterized by *Bellman's principle* [2], according to which the optimal action at time k is

$$\pi^*(b_0) = \arg\min_u \left\{c(b_0,u) + \mathrm{E}\left[J^*(b_1)|b_0,u\right]\right\},$$

where b_0 is the initial belief-state, b_1 is the random next belief-state, and $\mathrm{E}[J^*(b_1)|b_0,u]$ is the expected future cost of action u, which is also called the *expected cost-to-go* (ECTG). We assume a long horizon, which makes the dependence on the horizon of the ECTG negligible, and the optimal policy stationary.

Let us define the *Q-value* of taking action u at belief-state b as follows:

$$Q(b,u) = c(b,u) + \mathrm{E}\left[J^*(b')|b,u\right], \tag{5.4}$$

where b' is the random next belief-state. Therefore, the optimal policy is given by

$$\pi^*(b_0) = \arg\min_u Q(b_0, u).$$

In the decentralized case, we do not have access to the "global" belief-state. Instead, every agent maintains a *local* belief-state, which may vary from agent to agent (because the observation histories differ from agent to agent). Our approach is for each agent to compute its own local action as follows. The agent i computes, at time k,

$$\pi^i(b_k^i) = \arg\min_u Q(b_k^i, u), \tag{5.5}$$

where b_k^i is the local belief-state at agent i at time k. In other words, an agent i computes a joint action by taking into account only its local belief-state. After the computation of the joint action, agent i implements its local component. Our approach maintains the *look-ahead* (non-myopic) property that is a common theme among POMDP solution approaches, allowing us to account for the future impact of actions in our decision making.

In practice, it is intractable to compute the Q-value exactly. Therefore, the literature on POMDP methods has focused on approximation methods [1]. We extend one such method called *nominal belief-state optimization* (NBO) to solve our Dec-POMDP approximately. This method was introduced in [11] to solve a centralized UAV guidance problem, which was posed as a POMDP. The following subsection extends the NBO method to solve the current Dec-POMDP problem. In our description of the NBO method, as an approximation method for the function Q in (5.5), we will approximate the function J in (5.3) instead. This approximation to J is then optimized to approximate J^*, which is a part of function Q as indicated in (5.4).

5.5 NBO Approximation Method for Dec-POMDP

The computation complexity of the joint communication decisions at an agent i is exponential in the number of UAVs N and the length of the horizon H, which is prohibitive. For this reason, in the NBO method, we adopt a heuristic approach as follows. With regard

to the communication decisions, we let each agent optimize only its "local" communication decisions over the time horizon H. More precisely, we let an agent i optimize only its own communication decisions g_k^i (with whom to communicate) and l_k^i (what to communicate), along with the (global) joint kinematic controls over the time horizon H. We let each agent optimize the joint kinematic controls (along with local communication decisions) to induce coordination among the agents. After the computation of joint kinematic controls, an agent implements its local component at each time step. Therefore, we use the following objective function at agent i:

$$J(b_0^i) = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k^i, a_k, g_k^i, l_k^i)\middle| b_0\right],$$

where b_k^i is the local belief-state at time k, a_k is the joint action corresponding to the kinematic controls, (g_k^i, l_k^i) is the local communication decision, and $c(\cdot)$ is the local cost function that depends on the local belief-state, joint kinematic controls, and local communication decisions.

We assume that b_0 is Gaussian. Therefore, by (5.1) and (5.2), the local target belief state can be expressed (or approximated) as $b_k^{i,\chi}(\chi) = \mathcal{N}(\chi - \xi_k^i, \mathbf{P}_k^i)$, where ξ_k^i and $\mathbf{P}_k^i$ evolve according to the Kalman filter (or information filter) equations given the local observations and the observations received from other agents. The agent i, when it decides to communicate with agent j, sends its local target-observation generated at time $k-a$ ($a \in \{0, \ldots, L\}$) to agent j. At agent i, the local track state variables ξ_k^i and $\mathbf{P}_k^i$ evolve according to the Kalman filter equations given the locally generated target-observations and the target-observations (time-stamped) received from other agents.

In the NBO method, the objective function at agent i is approximated as follows:

$$J(b_0^i) \approx \sum_{k=0}^{H-1} c(\hat{b}_k^i, a_k, g_k^i, l_k^i),$$

where $\hat{b}_1^i, \hat{b}_2^i, \ldots, \hat{b}_{H-1}^i$ is a *nominal* local belief-state sequence. The nominal (local) target belief-state sequence can be identified with the nominal local tracks $(\hat{\xi}_k^i, \hat{\mathbf{P}}_k^i)$, which are obtained from the Kalman filter equations with exactly zero-noise (*nominal* noise) sequence

Algorithm 1 Information Filter for NBO

if $g_k^i \neq 0$ **then** ▷ (if i decides to communicate with g_k^i)
  if $l_k^i = k$ **then** ▷ (if i decides to send current observation to g_k^i)
    $\hat{\mathbf{P}}^i_{k+1|k} = \mathbf{F}\hat{\mathbf{P}}^i_k\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$
    $V \leftarrow (\hat{\mathbf{P}}^i_{k+1|k})^{-1}$
    $V \leftarrow \left[V + \mathbf{H}^{\mathrm{T}}\left[\mathbf{R}\left(\hat{\xi}^i_{k+1}, s^{g_k^i}_{k+1}\right)\right]^{-1}\mathbf{H}\right]$
  else ▷ (if i decides to send an observation from the recent past to g_k^i)
    Rollback the information filter at i to time step l_k^i and re-run the information filter algorithm with the assumption that the observation from agent g_k^i was available at time step l_k^i, and update $\hat{\mathbf{P}}^i_k$, and then do the following
    $\hat{\mathbf{P}}^i_{k+1|k} = \mathbf{F}\hat{\mathbf{P}}^i_k\mathbf{F}^{\mathrm{T}} + \mathbf{Q}$
    $V \leftarrow (\hat{\mathbf{P}}^i_{k+1|k})^{-1}$
  end if
end if
▷ (updating with the locally generated observation)
$V \leftarrow \left[V + \mathbf{H}^{\mathrm{T}}\left[\mathbf{R}\left(\hat{\xi}^i_{k+1}, s^i_{k+1}\right)\right]^{-1}\mathbf{H}\right]$
$\hat{\mathbf{P}}^i_{k+1} = V^{-1}$

as follows:

$$\hat{b}_k^{i,\chi}(\chi) = \mathcal{N}\left(\chi - \hat{\xi}^i_k, \hat{\mathbf{P}}^i_k\right),$$

$$\hat{\xi}^i_{k+1} = \mathbf{F}\hat{\xi}^i_k,$$

and the evolution of $\hat{\mathbf{P}}^i_k$ is described in Algorithm 1, where s^i_{k+1} evolves according to the UAV kinematic equations (defined earlier in Subsection 5.4.1) given the control commands.

In the NBO method, each agent i computes the path costs originating from i over the time horizon H. These path costs depend on the locations of the neighboring agents over the time horizon H. Each agent implements its control actions independently (i.e., there is no central controller), which means that an agent has no access to the future locations of its neighboring agents. Therefore, each agent computes the path costs over the horizon H by evolving (only for predictive-lookahead purposes) the locations of other agents according to the locally computed joint kinematic controls.

Given the communication decision (g_k^i, l_k^i) at agent i, we compute $\hat{\mathbf{P}}^i_k$ by assuming that the agent i receives an observation from agent g_k^i generated at time l_k^i. The rationale for doing this is as follows. When an agent i sends an observation from time step l_k^i to an agent

j, only agent j perceives the benefit of this action as the measurement fusion happens at agent j. The agent i, however, does not directly perceive the benefit from this measurement fusion at j because our system is decentralized. Therefore, the agent i computes the benefit (from measurement fusion) of sending the observation from time l_k^i to agent j approximately by assuming that agent j sends the observation generated at time l_k^i to agent i and that the measurement fusion happens at agent i. This idea is captured in the equations presented in Algorithm 1.

The NBO cost function, which includes the mean-squared error between the tracks and the targets and the cost of communication, can be written as

$$c(\hat{b}_k^i, a_k, g_k^i, l_k^i) = \operatorname{Tr} \hat{\mathbf{P}}_{k+1}^i + \beta\alpha \hat{D}_k^{i,g_k^i},$$

where β is a scaling factor, α is a given proportionality constant, and $\hat{D}_k^{i,g_k^i}$ is the length of the shortest path between agents i and g_k^i (obtained from Dijkstra's algorithm, where the length of each path is the path cost defined earlier) if there exists at least one path between i and g_k^i or $\hat{D}_k^{i,g_k^i}$ is a (relatively) large and constant value otherwise (i.e., when there exists no path). The function $\hat{D}_k^{i,g_k^i}$ is evaluated at agent i by evolving the locations of other UAVs with the locally computed joint kinematic controls. Therefore, the cumulative cost function at agent i is given by (with truncated horizon [11, 45, 46])

$$J_H(b_0^i) = \sum_{k=0}^{H-1} \left(\operatorname{Tr} \hat{\mathbf{P}}_{k+1}^i + \beta\alpha \hat{D}_k^{i,g_k^i} \right). \tag{5.6}$$

Here, we adopt an approach called "receding horizon control," according to which we optimize the action sequence for H time steps at the current time step and implement only the action corresponding to the current time step and again optimize the action sequence for H time steps in the next time step.

5.6 Simulation Results

We implement our approach in MATLAB, where we use the command *fmincon* (an optimization tool in MATLAB) to minimize the objective function in (5.6). The length of

the time horizon H is set to six time steps. The target measurement error, i.e., w_k^i in (5.2) is distributed according to the normal distribution $\mathcal{N}(0, \mathbf{R}_k(\chi_k, s_k^i))$, where $\mathbf{R}_k$ reflects 10% range standard deviation and 0.01π radian angular standard deviation. For the purpose of simulations, we set the value of α in (5.6) to be 0.01. In Figures 5.1, 5.2, 5.3, 5.6, and 5.7, the trajectory of a target is represented by a sequence of small circles and the trajectory of a UAV is represented by a curve joining the arrows that point toward the heading direction of the UAV.

We define the following performance metrics: 1) *average target-location error* and 2) *average communication cost*. The *average target-location error* is computed as follows. At every time step, we compute the squared distance (squared error) between the actual target location and the estimated target location from each UAV and we find the average of these errors (from each UAV's target location estimate). We summate these average errors over the simulation runtime; the mean of these average errors (from each Monte Carlo run) is called the *average target-location error*. The *average communication cost* is computed as follows. At every time step, based on who is communicating with whom, we summate (over each pair of communicating UAVs) the costs of these communications, and we call the output of this summation CommCost-per-step. At every step of the simulation runtime, we compute the CommCost-per-step, and the mean of these CommCost-per-steps (from each Monte Carlo run) is called the *average communication cost*.

We simulate a scenario with two UAVs and two targets as shown in Figures 5.1. In Figure 5.1, the two targets start at the bottom, and as time progresses, the right target moves toward the north-east and the left target moves toward the north-west. In this scenario, the value of β in (5.6) was set to be 1. It is evident from Figure 5.1 that the UAVs coordinate with each other, with the aid of communication, to track the targets that are moving away from each other (of course, this motion is not known to the UAVs beforehand). This figure shows one UAV following the right target and the other UAV following the left target, which demonstrates the coordination among the UAVs in maximizing the coverage of targets. In

summary, our approach induces coordination among the UAVs even with restricted communication (as in Section 5.3) and with relatively low computational time. Figures 5.2 and 5.3 depict the simulation of the above scenario with $\beta = 50$ and $\beta = 100$ in (5.6) respectively.

To provide insight into the computational complexity, we evaluate the average time it takes to compute the control commands for each UAV in MATLAB. The average computational time is 5.2 sec with $H = 6$ on a lab computer (Intel Core i7-860 Quad-Core Processor with 8MB Cache and 2.80 GHz speed). This computation time can be greatly reduced on a better processor and by further optimizing the code (e.g., by performing parallel computations).

Next, we conduct an empirical study to evaluate the affect of β on the performance with respect to *average target-location error* and *average communication cost*. We simulate the above scenario (with two UAVs and two targets) for 50 Mote-Carlo runs for $\beta = 1$, $\beta = 50$, and $\beta = 100$ in (5.6). For each β and in each Mote-Carlo run, we compute the *average target-location error* and *average communication cost*. Figure 5.4 shows the plots of the cumulative frequency of *average target-location error*s for each value of β. This plot demonstrates that the performance with respect to *average target-location error* degrades as the value of β increases, as expected. Figure 5.5 shows the plot of cumulative frequency of *average communication cost*s for each value of β. This plot demonstrates that the performance with respect to *average communication cost* improves as the value of β increases, again as expected. Therefore, we can use β as a tuning parameter to trade off between the performances with respect to *average target-location error* and *average communication cost*, which is evident from Figures 5.4 and 5.5.

5.6.1 Dec-POMDP Approach vs. Greedy Approach

In this subsection, we compare the performance of our Dec-POMDP approach with a greedy approach (defined as follows). In the greedy approach, the UAVs do not communicate with each other, and each UAV optimizes only its own kinematic controls over the time

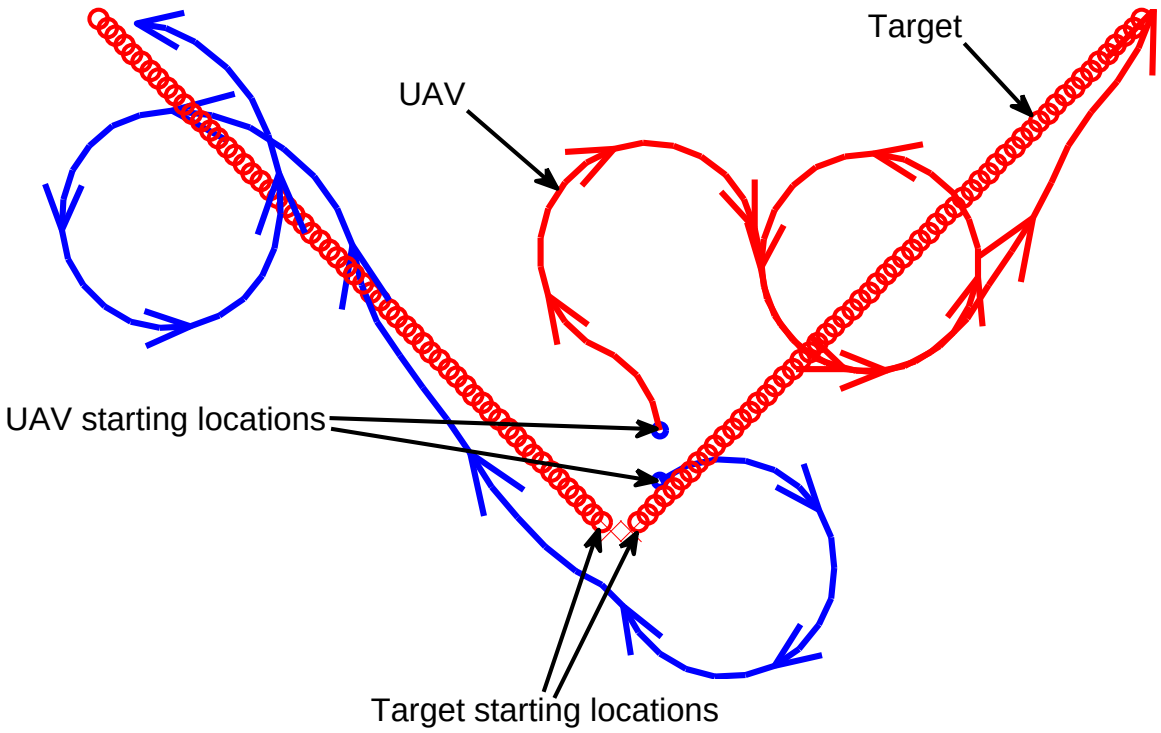


Figure 5.1: Two UAVs tracking two targets; $\beta = 1$

horizon ($H = 6$). Clearly, our Dec-POMDP approach induces cooperation among the UAVs by letting each UAV optimize the joint kinematic controls (along with local communication decisions), and the greedy approach is non-cooperative in the sense that each UAV behaves in a non-cooperative manner by optimizing only its own kinematic controls. We implement these two approaches for a scenario with two UAVs and two targets. Figures 5.6 and 5.7 depict the behavior of the UAVs in the Dec-POMDP and greedy approaches respectively. We run this simulation for 200 Monte Carlo runs, and compare the performances (with respect to *average target-location error*) of these two approaches, as depicted in Figure 5.8. Figure 5.8 demonstrates that our Dec-POMDP approach significantly outperforms the greedy approach. It is unsurprising that the Dec-POMDP approach outperforms the greedy approach. The purpose of Figure 5.8 is to show the quantitative difference in performance between these approaches. Specifically, the Dec-POMDP approach results in average target location-error

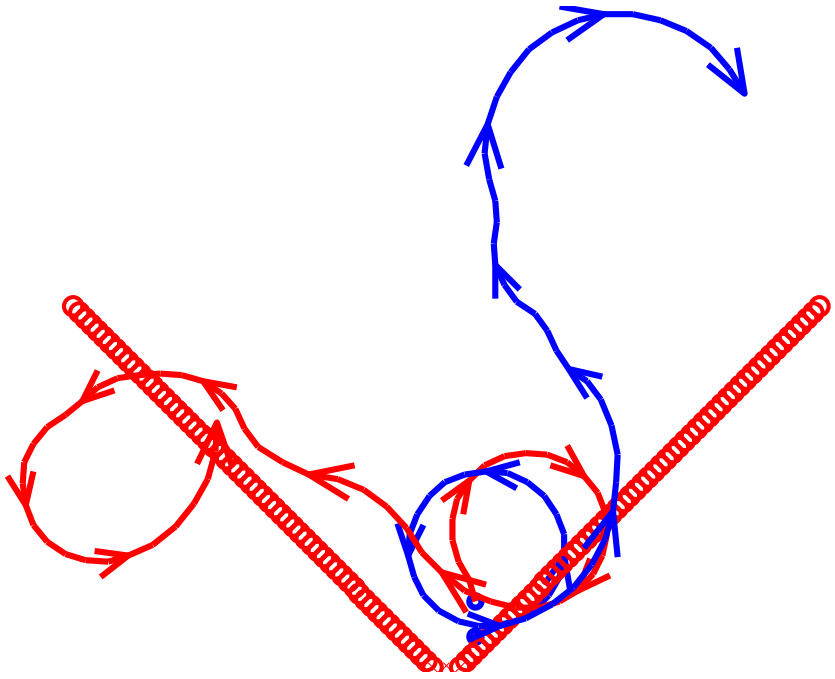


Figure 5.2: Two UAVs tracking two targets; $\beta = 50$

values that are approximately three times smaller compared to that of the greedy approach.

5.6.2 Optimized Communication Scheme vs. Fixed Communication Scheme

In our Dec-POMDP approach, we explicitly optimize the communication among the UAVs along with the joint kinematic controls. We now compare the performance of this approach with an approach where the communication scheme is fixed, i.e., not optimized. We call this new scheme the "fixed communication scheme." An overview of this scheme is given below.

Fixed Communication Scheme. In this scheme, we do not optimize the communication decisions, i.e., we only optimize the joint kinematic controls at each UAV. Each UAV, at each time step, sends the observations generated in the current time step to its closest neighbor among all the neighbors in its communication range. Since communication decisions cannot be controlled anymore, the action space of the Dec-POMDP includes only the kinematic controls of the UAVs. Moreover, the cost function that is used does not penalize

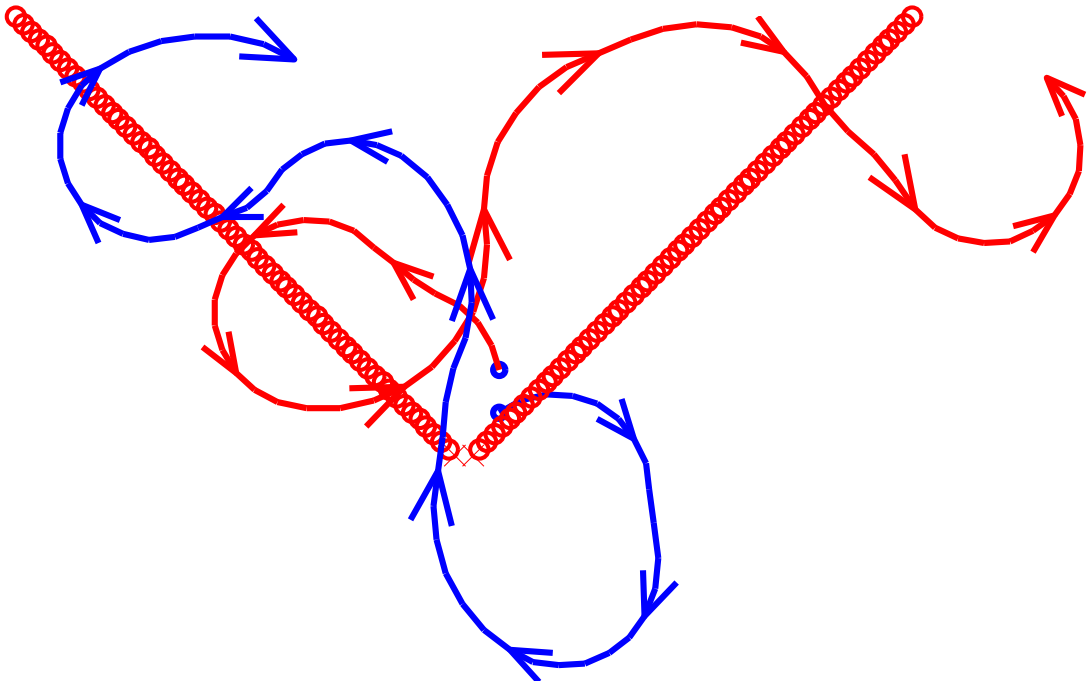


Figure 5.3: Two UAVs tracking two targets; $\beta = 100$

the communication (i.e., $\beta = 0$ in (5.6)). However, we incorporate the benefits (from measurement fusion) of this fixed communication scheme while computing trace objective in the NBO cost function in (5.6) (with $\beta = 0$).

We implement these two approaches for a scenario with three UAVs and two targets. Figure 5.9 depicts the performance comparison of the fixed communication scheme and the optimized communication scheme with respect to both the performance measures—*average target-location error* and *average communication cost*. It is evident from Figure 5.9 that the performance of the fixed communication scheme is slightly better than the optimized communication scheme with respect to *average target-location error*, which is to be expected because the fixed communication scheme does not use the communication cost in its cost function and is focused on minimizing only location error. However, the optimized communication scheme significantly outperforms the fixed communication scheme in terms of *average communication cost*.

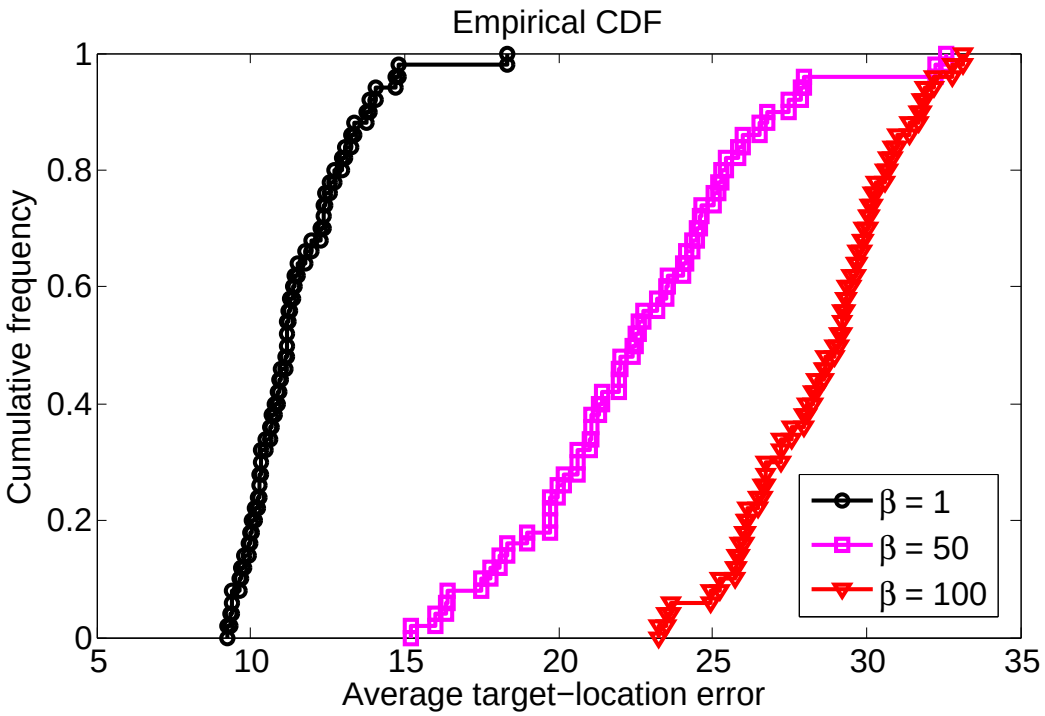


Figure 5.4: Performance with respect to *average target-location error* for various values of β

5.7 Concluding Remarks and Future Work

In this study, we designed a decentralized guidance control method for UAVs tracking multiple targets based on the theory of *decentralized partially observable Markov decision process* (Dec-POMDP). We extended a POMDP approximation method called *nominal belief-state optimization* (NBO) to solve our decentralized guidance control problem, which we posed as a Dec-POMDP. Although the communication between the UAVs was restricted, the NBO method achieved coordination among the UAVs, as evident from the simulation results in Section 5.6. The results also demonstrate that the parameter β can be used as a tuning parameter to trade off between the performances with respect to the *average target-location error* and the *average communication cost*. Specifically, when we increased the value of β (which is the weight on the cost of communication) the performance with respect to *average target-location error* (or tracking error) degraded while the performance with respect

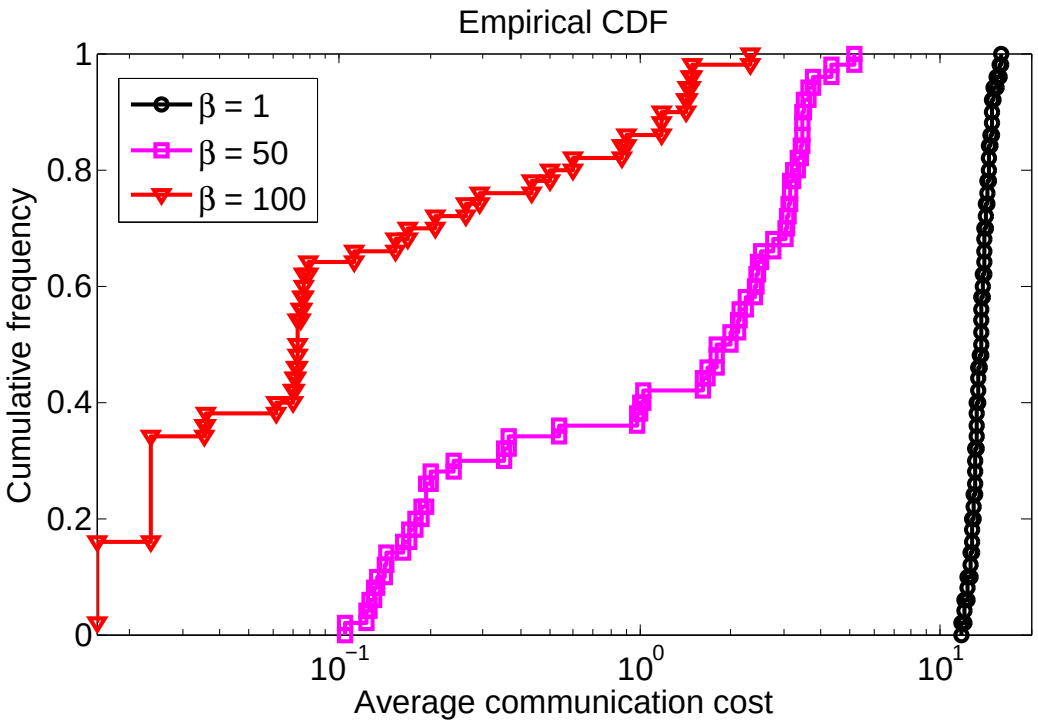


Figure 5.5: Performance with respect to *average communication cost* for various values of β

to *average communication cost* improved. In other words, the UAVs communicate less often at the expense of degraded tracking performance when we increase the value of β.

We then compared the performance of our Dec-POMDP approach with a greedy approach. In the Dec-POMDP approach, each UAV optimizes joint actions and implements its local component. In contrast, in the greedy approach, each UAV only optimizes its own kinematic controls. We showed quantitatively how much our Dec-POMDP approach outperforms (with respect to *average target-location error*) the greedy approach.

In our Dec-POMDP approach, we optimized the communication decisions along with the UAV kinematic controls by including the communication decisions in the Dec-POMDP action space. To demonstrate the effectiveness of this approach, we compared the performance of this optimized communication scheme with a different scheme called the fixed communication scheme. In this fixed communication scheme, each UAV communicates with its closest neighbor among all the neighbors in its communication range. Our results demonstrated

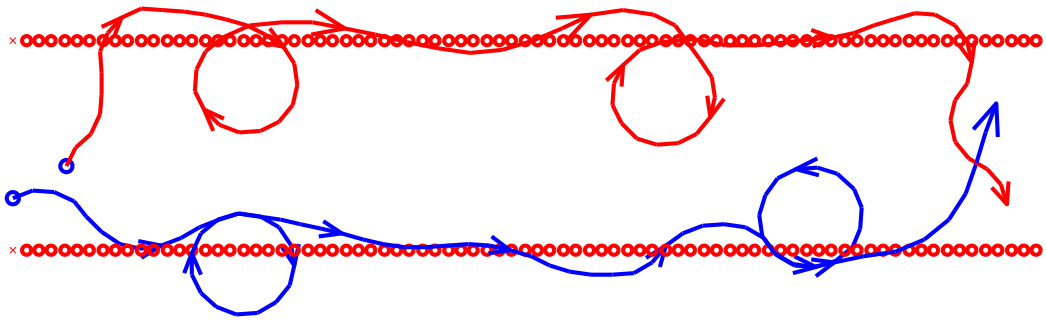


Figure 5.6: Two UAVs tracking two targets via Dec-POMDP approach

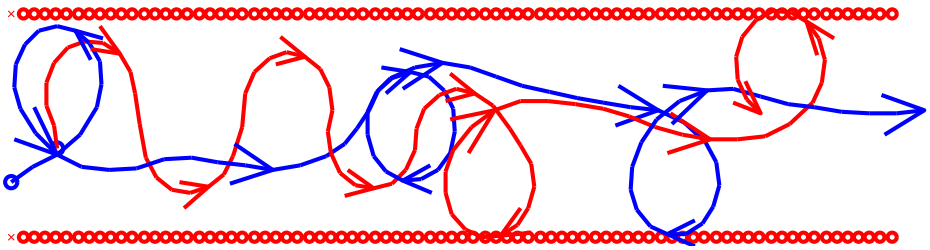


Figure 5.7: Two UAVs tracking two targets via Greedy approach

that the performance of the fixed communication scheme is slightly better than the optimized communication scheme with respect to the *average target-location error*, which was expected because the fixed communication scheme does not use the communication cost in its cost function and is focused on minimizing only location error. However, the optimized communication scheme significantly outperformed the fixed communication scheme in terms of *average communication cost*.

In our study, we did not incorporate communication delays, i.e., we assumed that communication delays are sufficiently small relative to the time duration between decision epochs that any information communicated at discrete-time k is received in time for decision-making at discrete-time $k+1$. It would be interesting to see how communication delays affect the performance of these decentralized UAV guidance algorithms. The results in this chapter were published in [53, 54].

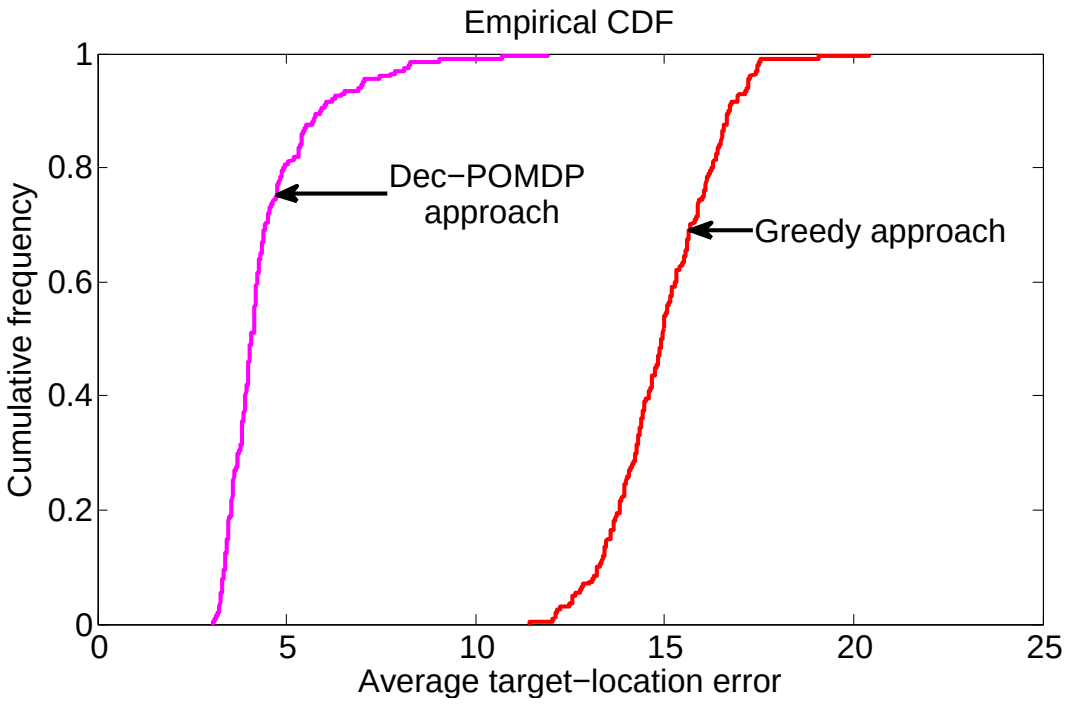


Figure 5.8: Dec-POMDP approach vs. greedy approach

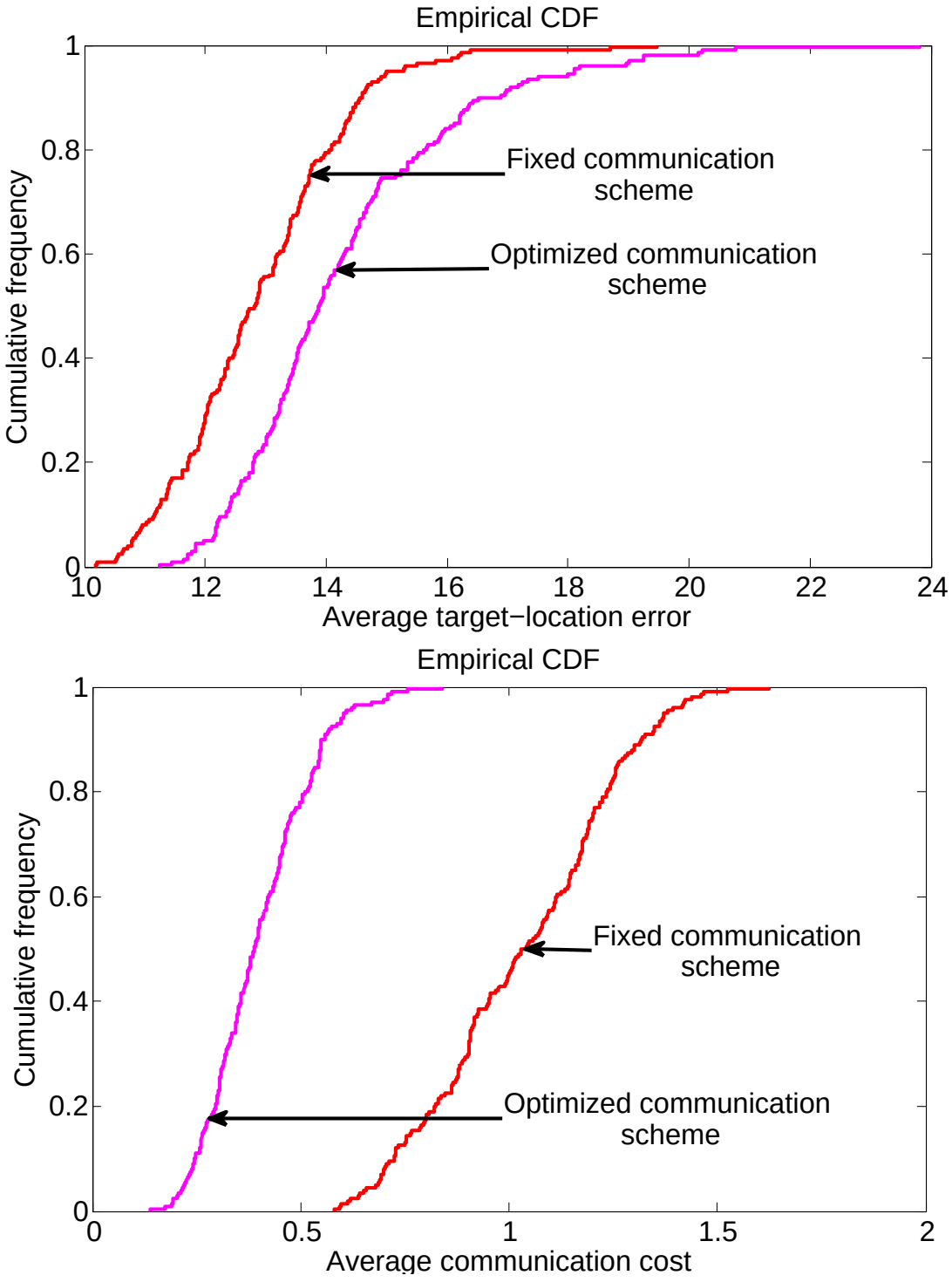


Figure 5.9: Fixed communication scheme vs. optimized communication scheme

CHAPTER 6

GUIDANCE OF AUTONOMOUS AMPHIBIOUS VEHICLES FOR FLOOD RESCUE SUPPORT

6.1 Introduction

Various guidance algorithms for autonomous amphibious vehicles (AAVs) are being designed and tested to fight today's global warming disasters such as flooding, typhoon, and hurricane [55–57]. With this motivation, we present a guidance framework to control multiple AAVs to rescue multiple victims (henceforth called targets) in a flood situation, where the flood water (interchangeably called river) flows along a valley as shown in Figure 6.1. A target is said to be rescued when an AAV is within the circular region of radius $d_{\text{dist-thresh}}$ on the 2-D plane around the target. In general, AAVs are equipped with various advanced sensors such as polarized stereo vision, laser scanning, and SONAR [58–60]. The sensors on-board an AAV generate the (noisy) measurements corresponding to the targets and the river. Our goal is to design a path-planning algorithm that guides the AAVs so that every target gets rescued, while maximizing a performance measure (discussed later). The algorithm runs on a notional central fusion node, which collects the measurements from the sensors on-board each AAV, fuses them and updates the tracks on the targets and the river state (discussed later), computes the control commands for the AAVs, and sends the control commands back to the AAVs.

Guidance control methods [55, 61–63] for AAVs are normally based on a standard three-layered system architecture that requires human-machine interactions. We design the guidance algorithm based on the theory of *partially observable Markov decision process* (POMDP) [11, 45]. There are several other autonomous control methods in the literature for AAVs and

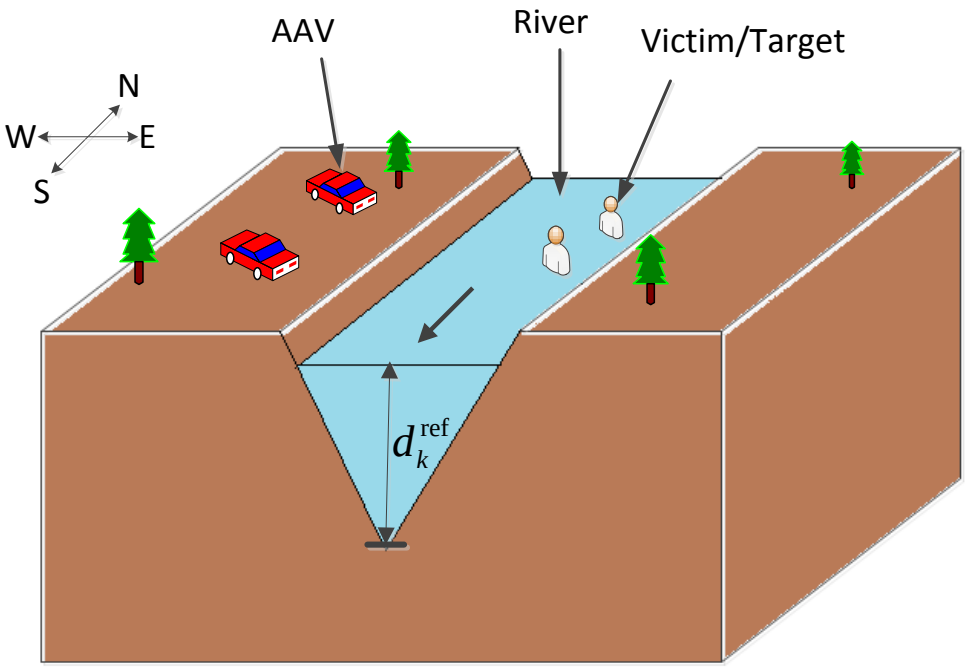


Figure 6.1: Flood Scenario

underwater vehicles, e.g., [64–66]. Our approach differs from these existing approaches in that we place the guidance problem in the context of POMDP, wherein this approach has a look-ahead property, which trades off short-term for long-term performance.

6.2 Problem Specification

The AAV guidance problem is specified as follows:

- **Targets**. In this study, we assume that there are multiple mobile targets (flood victims) located in a river, being drifted down by the flood water, as shown in Figure 6.1.

- **Autonomous Amphibious Vehicles (AAVs)**. There are multiple autonomous amphibious vehicles (AAVs) located on the shore, as shown in Figure 6.1. An AAV is controlled by the following kinematic controls: forward acceleration and steering angle. Each AAV is equipped with on-board sensors that generate measurements of targets

and the river depth. In this problem, AAVs float when moving in the river. For the purpose of this study, we assume that the number of AAVs and the number of targets is the same.

- **Environmental Conditions**. The elevation map of the region is known a priori. The landscape for this problem is shown in Figure 6.1, which shows a river flowing along a valley from the north toward the south. The state of the river includes the depth d_k^{ref} at a reference point on the map (lowest point in the landscape, e.g., some location at the bottom of the valley as shown in Figure 6.1).

- **River Model**. Typically a river flows slowly near the coastlines (where the river is shallow), and flows quickly far from the coastlines (i.e., toward the center of the river where the river is deep). In this chapter, we assume that the river flows from the north toward the south in a v-shaped channel as shown in Figure 6.1. We adopt the *logarithmic velocity profile* to model the velocity of the flow (see [67] for a detailed description). According to this model, the speed of the river, at the surface, at the location (p, q) at time k is given by

$$w_k(p, q) = C_1 \left[\log(d_k(p, q)) + C_2\right],$$

  where $d_k(p, q)$ is the depth of the river at the location (p, q) at time k, and C_1 (a function of the viscosity and the density of flood water) and C_2 are constants (see [67] for more details).

- **Observations**. The sensors on-board an AAV generate noisy observations of target locations and the depth of the river directly beneath the vehicle, i.e., the sensors generate the observations of the depth of the river only when the AAV is in the river.

- **Objective**. A target is said to be rescued if there is an AAV within a circular region of radius $d_{\text{dist-thresh}}$ around the target. The objective is to minimize the average rescue

time, where the average is over the number of targets, and the rescue time of a target is defined as the time it takes to rescue the target.

6.3 Problem Formulation

We cast the AAV guidance problem into the framework of a *partially observable Markov decision process* (POMDP). A POMDP is a mathematical framework useful for solving resource control problems, and enables us to exploit approximation methods for POMDPs to design our AAV guidance algorithm. A POMDP evolves in discrete time-steps. We use k as the discrete-time index. To cast the AAV guidance problem into the POMDP framework, we need to define the following key components in terms of our guidance problem as follows.

States. Let x_k represent the state of the system at time k. The state of the system includes the state of the vehicles (AAVs) s_k, river state (depth of the river at a reference location) d_k^{ref}, target state χ_k, and track states $(\xi_k^{\text{riv}}, P_k^{\text{riv}}, \xi_k^{\text{targ}}, P_k^{\text{targ}})$, i.e., $x_k = (s_k, d_k^{\text{ref}}, \chi_k, \xi_k^{\text{riv}}, P_k^{\text{riv}}, \xi_k^{\text{targ}}, P_k^{\text{targ}})$. The vehicle state s_k includes the locations and the velocities of the AAVs at time k. The river state d_k^{ref} is the depth of the river at the reference point at time k. The reference point is the lowest point in the elevation map, i.e., some location at the bottom of the valley in the landscape, as shown in Figure 6.1. Here we assume that the flow direction of the river is the same everywhere and is known a priori. The target state χ_k includes the locations and the velocities of the targets at time k. The track states represent the state of the tracking algorithm, where ξ_k^{riv} and P_k^{riv} are the mean and the variance, standard in Kalman filter equations, corresponding to the river state, and similarly, ξ_k^{targ} is the mean vector and P_k^{targ} is the covariance matrix corresponding to the target state.

Observations and Observation Law. The vehicle and the track states are assumed to be fully observable. The river and the target states are only partially observable. The observation of the river state at an AAV is given by

$$z_k^{\text{riv}} = \begin{cases} d_k^{\text{ref}} + n_k^{\text{riv}} & \text{if AAV is in river} \\ \text{no measurement} & \text{otherwise,} \end{cases} \tag{6.1}$$

where $n_k^{\text{riv}} \sim \mathcal{N}(0, R_k)$, and R_k is the measurement variance. The sensors at an AAV generate the measurement of the river-state only when the AAV is in the river. In practice, the sensors on an AAV measure the depth of the river exactly below the AAV. We wrote the observation model (6.1) as if the sensors are generating the observations of the depth of the river at the reference point. The rationale behind this assumption is that we can always calculate the depth of the river at the reference point given the elevation map and the observed depth of the river at a different location. The observation of the jth target at an AAV is given by

$$z_k^{\chi^j} = \begin{cases} H\chi_k^j + n_k^{\text{targ}} & \text{if there is} \\ & \text{line-of-sight,} \\ \text{no measurement} & \text{otherwise,} \end{cases}$$

where H is the target-state observation model, χ_k^j is the state of jth target, and $n_k^{\text{targ}} \sim \mathcal{N}(0, S_k)$ where S_k is the measurement covariance matrix. The line-of-sight between the target and the AAV is blocked sometimes, e.g., whenever the target sinks in the water.

Actions. The actions include the controllable aspects of the system. In this problem, the actions include the decisions on the assignment of AAVs to targets, and kinematic control commands for AAVs. Let u_k be the action tuple at time k, which is given by $u_k = (g_k, a_k)$, where a_k represents kinematic control vectors (includes forward acceleration and steering angle for each AAV), and g_k is a vector, which represents the assignment of AAVs to targets, i.e., $g_k(i) = j$ means that the ith AAV is assigned to the jth target. For the purpose of this study, the number of AAVs and the targets is the same. Each AAV is assigned to only one target, and each target gets assigned only one AAV, i.e., g_k represents a one-to-one correspondence between the AAVs and the targets.

State-Transition Law. The state-transition law specifies the next-state distribution given the current state and the action. The transition function for the vehicle state is given by: $s_{k+1} = \psi(s_k, a_k, \xi_k^{\text{riv}})$, where ψ (defined later) represents the AAV kinematic model, s_k is the vehicle state, a_k is the kinematic control vector (includes forward acceleration and steering angle), and ξ_k^{riv} is the estimated river state at time k. The river state evolves

according to the following equation:

$$d_{k+1}^{\text{ref}} = d_k^{\text{ref}} + o_k, \quad o_k \sim \mathcal{N}(0, U_k^{\text{riv}}),$$

where U_k^{riv} is the process variance corresponding to the river state evolution. The target state evolves according to

$$\chi_{k+1} = F\chi_k + e_k, \quad e_k \sim \mathcal{N}(0, U_k^{\text{targ}}), \tag{6.2}$$

where F represents the target motion model, and U_k^{targ} is the process covariance matrix corresponding to the target state evolution. The track states evolve according to the Kalman filter equations given the observations from the sensors on-board the AAVs. When the observations are not available, the track states evolve according to the Kalman filter equations, where only the *prediction step* is performed and the *update step* is not performed.

Cost. The cost function represents the cost of performing an action at the current state. The cost function is given by

$$C(x_k, u_k) = \sum_{i=1}^{N} \mathbb{1}\left\{\mathrm{E}\left[\left\| s_{k+1}^{i,\text{pos}} - \xi_{k+1}^{g_k(i),\text{targ,pos}} \right\| \,\middle|\, x_k, u_k\right] > d_{\text{dist-thresh}}\right\},$$

where $s_{k+1}^{i,\text{pos}}$ represents the 2-D position coordinates of ith AAV, and $\xi_{k+1}^{j,\text{targ,pos}}$ represents the estimated 2-D position coordinates of the jth target at time $k+1$, $||\cdot||$ is the Euclidean norm (everywhere in this chapter), and $\mathbb{1}\{\cdot\}$ is the indicator function which equals 1 when the expected distance between the AAV and the target at time $k+1$ is greater than some threshold distance $d_{\text{dist-thresh}}$ and 0 otherwise.

Belief State. The belief state b_k is the posterior distribution of the state at time k. The vehicle and the track states are assumed to be fully observable, i.e., the belief-state corresponding to the vehicle state is given by $b_k^s(s) = \delta(s - s_k)$, where $\delta(\cdot)$ is the Kronecker delta function. Similarly, the belief-states corresponding to the track states can be written in terms of the actual track states. The belief-states corresponding to the river and the target are the posterior distributions of d_k^{ref} and χ_k respectively given the history of observations.

6.4 Objective and Optimal Policy

The goal is to find the action sequence $(u_0, u_1, \ldots, u_{H-1})$ such that the expected cumulative cost over a time horizon H is minimized. The expected cumulative cost is given by

$$J_H = \mathrm{E}\left[\sum_{k=0}^{H-1} C(x_k, u_k)\right].$$

We can write the expected cumulative cost in terms of the belief states given the initial belief-state b_0 (similar to the treatment in [11, 45]) as follows:

$$J_H(b_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k, u_k) \,\middle|\, b_0\right],$$

where $c(b_k, u_k) = \int C(x, u_k) b_k(x)\, \mathrm{d}x$ and b_0 is the belief state at time $k = 0$. From Bellman's principle of optimality [2] the optimal objective function value is given by

$$J_H^*(b_0) = \min_u \left\{ c(b_0, u) + \mathrm{E}\left[J_{H-1}^*(b_1) \,\middle|\, b_0, u\right]\right\},$$

where b_1 is the random next belief state, J_{H-1}^* is the optimal cumulative cost over the horizon $H-1$, $k = 1, 2, \ldots, H-1$, and $\mathrm{E}[\cdot|b_0, u]$ is the conditional expectation given the current belief state b_0 and the current action u at time $k = 0$. Let us define the Q-value of taking action u given the current belief state b_0:

$$Q_H(b_0, u) = c(b_0, u) + \mathrm{E}\left[J_{H-1}^*(b_1) \,\middle|\, b_0, u\right]. \tag{6.3}$$

The optimal policy (from Bellman's principle) at time $k = 0$ can be written as

$$\pi_0^*(b_0) = \arg\min_u \, Q_H(b_0, u).$$

In general, it is hard to obtain the Q-value exactly. There are several approximation methods in the literature: heuristic expected-cost-to-go (ECTG) [68], parametric approximation [4], policy rollout [6], hindsight optimization [69], and foresight optimization [9]. In this chapter, we use one such approximation method called *nominal belief-state optimization* (NBO), which was introduced in [11] along with other approximations and techniques specific to

guidance problems. The rationale behind choosing NBO method over other methods to solve POMDP is that it is relatively inexpensive in terms of computation time, i.e., the computational requirements are not prohibitive unlike other approximation methods. The following subsection provides a brief description of the NBO method.

6.4.1 NBO Approximation Method

The computational requirements of obtaining the optimal assignments of AAVs to targets (g_k) over a long horizon is prohibitive. Also, we expect that the optimal assignment of AAVs to targets (g_k) over a long horizon does not change with time. For these reasons, in the NBO method, we keep the assignment of AAVs to targets fixed. In other words, in approximating the expected cost-to-go in (6.3), g_k remains fixed over the planning horizon H. Therefore, we drop the subscript k from g_k in the objective function used in the planning based on (6.3), i.e., $g_k = g$ for all k. In the NBO approximation method, we use the following objective function, written in terms of belief states:

$$J_H(b_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} c(b_k, a_k, g) \,\middle|\, b_0\right],$$

where a_k represents the kinematic controls for the AAVs, and g is the assignment of AAVs to the targets.

The belief-states corresponding to the river state and the target state are given by

$$b_k^{\mathrm{riv}}(d) = \mathcal{N}\left(d - \xi_k^{\mathrm{riv}}, P_k^{\mathrm{riv}}\right),$$
$$b_k^{\mathrm{targ}}(\chi) = \mathcal{N}\left(\chi - \xi_k^{\mathrm{targ}}, P_k^{\mathrm{targ}}\right),$$

where $(\xi_k^{\mathrm{riv}}, P_k^{\mathrm{riv}}, \xi_k^{\mathrm{targ}}, P_k^{\mathrm{targ}})$ are the track states corresponding to the river and the target states respectively, which evolve according to the Kalman filter equations. In the NBO method, we approximate the objective function as follows:

$$J_H(b_0) \approx \sum_{k=0}^{H-1} c(\hat{b}_k, a_k, g),$$

where $\hat{b}_1, \ldots, \hat{b}_{H-1}$ is a *nominal* belief-state sequence and the optimization is over an action sequence $g, a_0, \ldots, a_{H-1}$. We obtain the *nominal* belief-states by evolving the current belief-state with exactly zero-noise sequence over the horizon H (similar to the treatment in [11] and [45]). Therefore, the objective function from the NBO method is given by:

$$J_{\mathrm{NBO}}(b_0) = \sum_{k=0}^{H-1} \sum_{i=1}^{N} \mathbb{1}\left\{ \left\| \hat{s}_{k+1}^{i,\mathrm{pos}} - \hat{\xi}_{k+1}^{g(i),\mathrm{targ,pos}} \right\| > d_{\text{dist-thresh}} \right\},$$

where $\hat{s}_{k+1}^{i,\mathrm{pos}}$ is the *nominal* position of the ith AAV (defined below), $\mathcal{N}(\hat{\xi}_{k+1}^{j,\mathrm{targ}}, \hat{P}_{k+1}^{j,\mathrm{targ}})$ is the nominal belief-state of the jth target at time $k+1$, where $\hat{\xi}_{k+1}^{j,\mathrm{targ,pos}}$ (component of $\hat{\xi}_{k+1}^{j,\mathrm{targ}}$) represents the position estimate of the target. This nominal target belief-state is obtained by evolving the track state component $\hat{\xi}_k^{j,\mathrm{targ}}$ with exactly zero-noise sequence as follows:

$$\hat{\xi}_{k+1}^{j,\mathrm{targ}} = F \hat{\xi}_k^{j,\mathrm{targ}}.$$

The evolution of vehicle state depends on the river state estimate ξ_k^{riv}. In the NBO method, ξ_k^{riv} is replaced with $\hat{\xi}_k^{\mathrm{riv}}$ in the AAV kinematic model $\psi(\cdot)$, where $(\hat{\xi}_1^{\mathrm{riv}}, \ldots, \hat{\xi}_H^{\mathrm{riv}})$ are the *nominal* track state components corresponding to the river state, and the obtained positions of the ith AAV $\hat{s}_{k+1}^{i,\mathrm{pos}}$ are called *nominal positions*.

Here, we adopt an approach called "receding horizon control," according to which we optimize the action sequence for H time steps at the current time-step, implement only the action corresponding to the current time-step, and again optimize the action sequence for H time-steps in the next time-step. The length of the planning horizon H should be large enough for an AAV to receive a benefit by moving toward a target. Due to computational constraints, we cannot have an arbitrarily long horizon. Therefore we truncate the length of the horizon to a few time-steps (we set $H = 12$ in our simulations), and append the cost function with an appropriate *expected cost-to-go* (ECTG). The following is a distance-based ECTG:

$$J_H^{\text{dist-ECTG}} = \sum_{i=1}^{N} \left\| \hat{s}_H^{i,\mathrm{pos}} - \hat{\xi}_H^{g(i),\mathrm{targ,pos}} \right\|,$$

where $\hat{s}_H^{i,\mathrm{pos}}$ is the *nominal* position of the ith AAV and $\hat{\xi}_H^{j,\mathrm{targ,pos}}$ is the estimated location of the jth target (from NBO approach) at time $k = H$. Therefore, the objective function from

the NBO method is given by

$$J_{\text{NBO}}(b_0) = \sum_{k=0}^{H-1} \sum_{i=1}^{N} \mathbb{1}\left\{ \left\| \hat{s}_{k+1}^{i,\text{pos}} - \hat{\xi}_{k+1}^{g(i),\text{targ,pos}} \right\| > d_{\text{dist-thresh}} \right\} + J_H^{\text{dist-ECTG}},$$

where $J_H^{\text{dist-ECTG}}$ is the distance-based ECTG.

6.4.2 AAV Kinematics

The kinematic equations of an AAV vary depending on whether the AAV is in the river or on the land. When the AAV is in the river, we take into account the speed of the river to write the kinematic equations. The steering and thrust generation of the vehicle is modeled based on the work done by the authors of [56, 70], which is designed using single drive system. The vehicle is front-wheel driven on land. When the AAV is in the river, it is propelled using the centrifugal pump from the front wheels. The following subsections describe the kinematics of AAV on the land and in the river.

6.4.2.1 Kinematics of AAVs on the Land

This subsection provides the definition of ψ, which was introduced in Section 6.3, when the vehicle is on land. Let $s_k = (p_k, q_k, v_k, \theta_k)$ be the state of the vehicle at time k, where (p_k, q_k) represents the location of the vehicle on the 2-D plane, v_k represents the speed of the vehicle along the heading direction, θ_k represent the heading angle of the vehicle at time k. Let $a_k = (f_k, \phi_k)$ represent the action vector of the vehicle, where f_k represents the acceleration along the direction of the front wheels and ϕ_k represents the steering angle of front wheels. The (simplified) schematic of a basic four-wheeled vehicle is shown in Figure 6.2. The control variable f_k lies within the interval $[-f_{\text{land}}, f_{\text{land}}]$, where f_{land} (or $-f_{\text{land}}$) is the maximum acceleration (or deceleration), and the control variable ϕ_k lies within the interval $[-\delta_{\text{land}}, \delta_{\text{land}}]$, where δ_{land} is the maximum steering angle. The function ψ can be

specified by a set of non-linear kinematic equations, as shown below:

$$\begin{aligned} p_{k+1} &= p_k + v_k T \cos(\theta_k), \\ q_{k+1} &= q_k + v_k T \sin(\theta_k), \\ v_{k+1} &= v_k + f_k T \cos(\phi_k), \\ \theta_{k+1} &= \theta_k - \frac{2 f_k T^2 L}{W^2 + L^2} \sin(\phi_k), \end{aligned} \tag{6.4}$$

where T is the length of the time-step, W is the width of the vehicle, and L is the distance between the front-axle and the rear-axle. The derivation of the heading angle update (6.4) is as follows. When the front-wheels of the vehicle are oriented at a particular angle ϕ_k with respect to the main axis of the vehicle (as shown in Figure 6.2), the heading direction of the vehicle at time $k+1$ is derived as follows:

$$\begin{aligned} \alpha &= \arctan(W/L), \\ \theta_{k+1} &= \theta_k + \frac{T^2}{\sqrt{L^2 + W^2}} (f^{\theta}_{k,1} - f^{\theta}_{k,2}), \\ &= \theta_k + \frac{f_k T^2}{\sqrt{L^2 + W^2}} [\sin(\alpha - \phi_k) - \sin(\alpha + \phi_k)], \\ &= \theta_k - \frac{2 f_k T^2}{\sqrt{L^2 + W^2}} [\cos(\alpha) \sin(\phi_k)], \\ &= \theta_k - \frac{2 f_k T^2 L}{W^2 + L^2} \sin(\phi_k). \end{aligned}$$

6.4.2.2 Kinematics of AAVs on the River

This subsection provides the definition of ψ, when the vehicle is in the river. The kinematic equations of the AAV motion are as follows:

$$\begin{aligned} p_{k+1} &= p_k + v_k T \cos(\theta_k) + \hat{w}^x_k(p_k, q_k) T, \\ q_{k+1} &= q_k + v_k T \sin(\theta_k) + \hat{w}^y_k(p_k, q_k) T, \end{aligned}$$

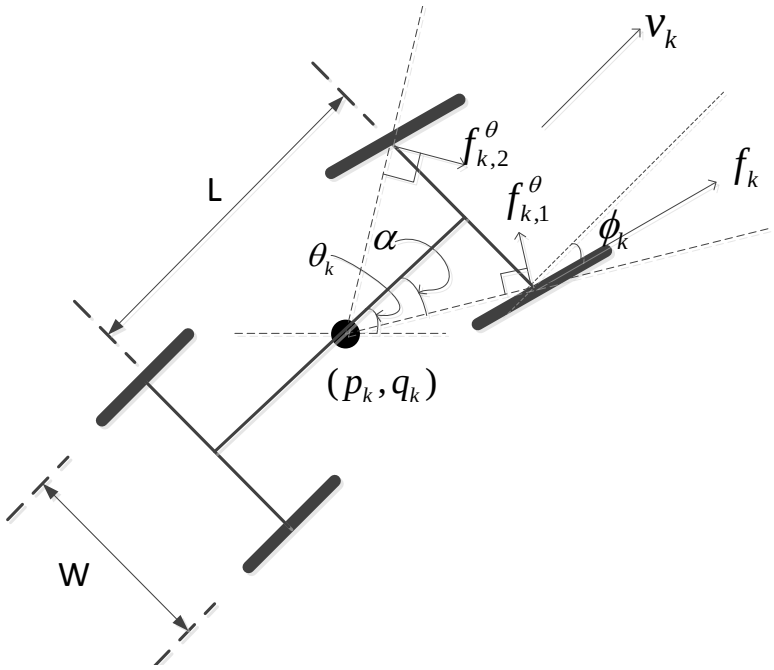


Figure 6.2: Free body diagram of an AAV

where $\hat{w}^x_k(p_k, q_k)$ and $\hat{w}^y_k(p_k, q_k)$ are the estimated speeds of the river at the location (p_k, q_k) in x and y directions respectively, which are obtained from the river state estimate $\hat{\xi}^{\text{riv}}_k$ and the river model presented in Section 6.2. The speed and the heading angle update equations remain the same as in the case of land. When in water (or river), the control variable f_k lies within the interval $[-f_{\text{water}}, f_{\text{water}}]$, where f_{water} is the maximum acceleration, and ϕ_k lies within the interval $[-\delta_{\text{water}}, \delta_{\text{water}}]$, where δ_{water} is the maximum steering angle. Typically, the values of f_{water} and δ_{water} are much smaller compared to that of f_{land} and δ_{land}.

6.5 Simulation

We implement the NBO method in MATLAB, and we use the command *fmincon* (MATLAB's optimization tool) to solve the optimization problem. For performance comparison, we also implement a greedy approach, where we pick uniformly at random one of the $N!$ possible assignments of N AAVs to N targets at the start of the simulation and this assignment is fixed throughout the simulation. In the greedy approach, we optimize only the

current kinematic controls for an AAV such that its distance from the estimated location of its assigned target, in the next time-step, is minimized. Our simulation environment is two dimensional, i.e., the AAVs, the river, and the targets move on a 2-D plane. According to the river model, the speed of the river stream w_k at a location (p, q) on the 2-D plane is given by $w_k = C_1[\log(d_k(p, q)) + C_2]$, where $d_k(p, q)$ is the depth of the river at (p, q), and C_1 and C_2 are constants. Since the depth of the river is not fully observable, we estimate $d_k(p, q)$ as follows. The elevation map of the landscape is known a priori, i.e., if we know the depth of the river at a particular location, we can obtain the depth of the river at all locations. Therefore, we estimate the depth of the river at location (p, q), i.e., $\hat{d}_k(p, q)$ using the estimated depth of the river at the reference point $\hat{d}_k^{\text{ref}}$ $(= \hat{\xi}_k^{\text{riv}})$. Therefore, the estimated speed of the river at location (p, q) is given by $\hat{w}_k(p, q) = C_1[\log(\hat{d}_k(p, q)) + C_2]$. We set the length of the horizon H to 12 time-steps, and the length of the times-step T to 1 second. In the simulations, the flooded river flows along a valley in the landscape from the north toward the south as shown in Figure 6.1. Since the simulations are in 2-D, the river flows toward the $-y$ direction, and the river speed in x direction (toward the east) is zero at every location. Therefore, the estimated speeds of the river at location (p, q) in x and y directions are given by $\hat{w}_k^x(p, q) = 0$ and $\hat{w}_k^y(p, q) = -C_1[\log(\hat{d}_k(p, q)) + C_2]$. Here, we model the dynamics of the target motion by the *constant velocity model* (see [33] for the definition of the variables F and U_{targ} in (6.2)).

In the simulations, an AAV is represented by a rectangle, and the line connecting the rectangles represents the trajectory of the AAV. We define a performance metric called *average rescue time*—the average of the rescue times for each target (the rescue time of a target is the time elapsed after the start of the simulation until it is rescued). The POMDP cost function defined in Section 6.3 is reflective of this performance metric. We simulate two scenarios: Scenario I and Scenario II. In Scenario I, there are two AAVs, each one located on the opposite banks of the river, and two targets are moving (being drifted by the moving water) in the river, as shown in Figure 6.3. Figure 6.3 shows a snapshot of the

scenario at the end of the simulation with the NBO approach, where the *average rescue time* is 25.5 time-steps. We also simulate Scenario I with the greedy approach, as shown in Figure 6.4, where the *average rescue time* is 40.5 time-steps. In Scenario II, there are two AAVs on the left bank of the river, and two targets are moving in the river. We simulate this scenario with both the NBO and the greedy approaches. Figure 6.5 shows the snapshot of the scenario with the NBO approach at the end of the simulation, where the *average rescue time* is 42 time-steps, and Figure 6.6 shows the simulation of the same scenario with the greedy approach, where the *average rescue time* is 62.5 time-steps. The simulation of these two scenarios demonstrates that the NBO approach achieves a better coordination among the AAVs compared to the greedy approach while rescuing the targets, as evident from the *average rescue time*s.

We compare the performance of the NBO approach with that of the greedy approach through Monte-Carlo simulations. We simulate Scenario I and Scenario II with the NBO and the greedy approaches separately for 50 Monte-Carlo runs. In each scenario, we compute the *average rescue time* in every run for both the NBO and the greedy approaches. Figures 6.7 and 6.8 show the plots of the cumulative frequencies of *average rescue time*s for the NBO and the greedy approaches for Scenarios I and II respectively. Figures 6.7 and 6.8 demonstrate that the NBO approach significantly outperforms the greedy approach.

6.6 Conclusions and Remarks

We designed a guidance algorithm for autonomous amphibious vehicles (AAVs) to rescue moving targets in a 2-D flood scenario, where the flood water flows across the scene and the targets move in the flood water. We designed this algorithm based on the theory of *partially observable Markov decision process* (POMDP). Since a POMDP problem is intractable to solve exactly, we used an approximation method called *nominal belief-state optimization* (NBO). We simulated a few scenarios with two AAVs and two targets to demonstrate the coordination among the AAVs achieved by the NBO approach. We defined a performance

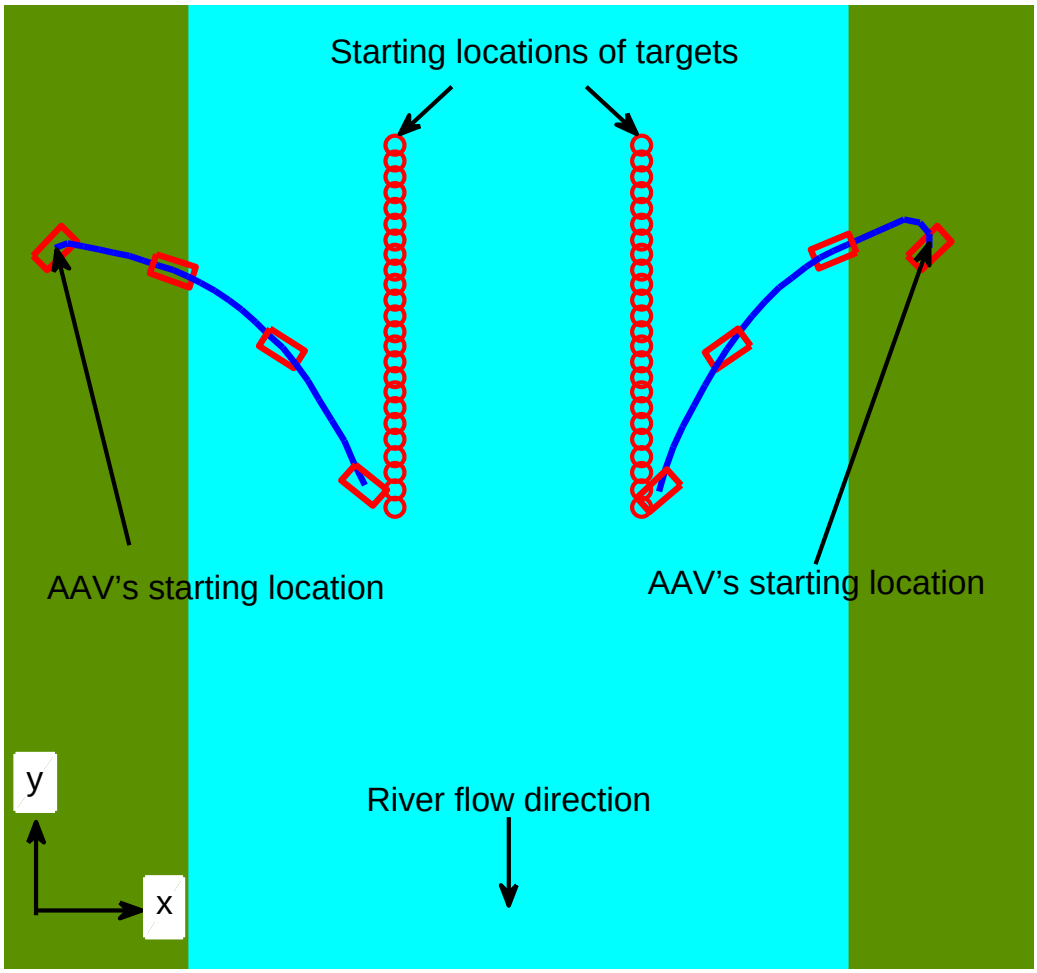


Figure 6.3: Simulation of Scenario I with NBO approach, *average rescue time*=25.5 steps

metric called *average rescue time* to compare the performance of our approach with a greedy approach. Our results show that the NBO approach outperforms the greedy approach significantly. This was expected because, unlike the greedy approach, the NBO approach has a lookahead property, i.e., the NBO approach trades off the short-term performance for the long-term performance. Although the greedy approach achieves coordination among the AAVs in that the AAVs eventually rescue all the targets, but the performance in terms of *average rescue time*, which is crucial in these kind of rescue missions, is poor compared to our NBO approach. The results in this chapter were published in [71, 72].

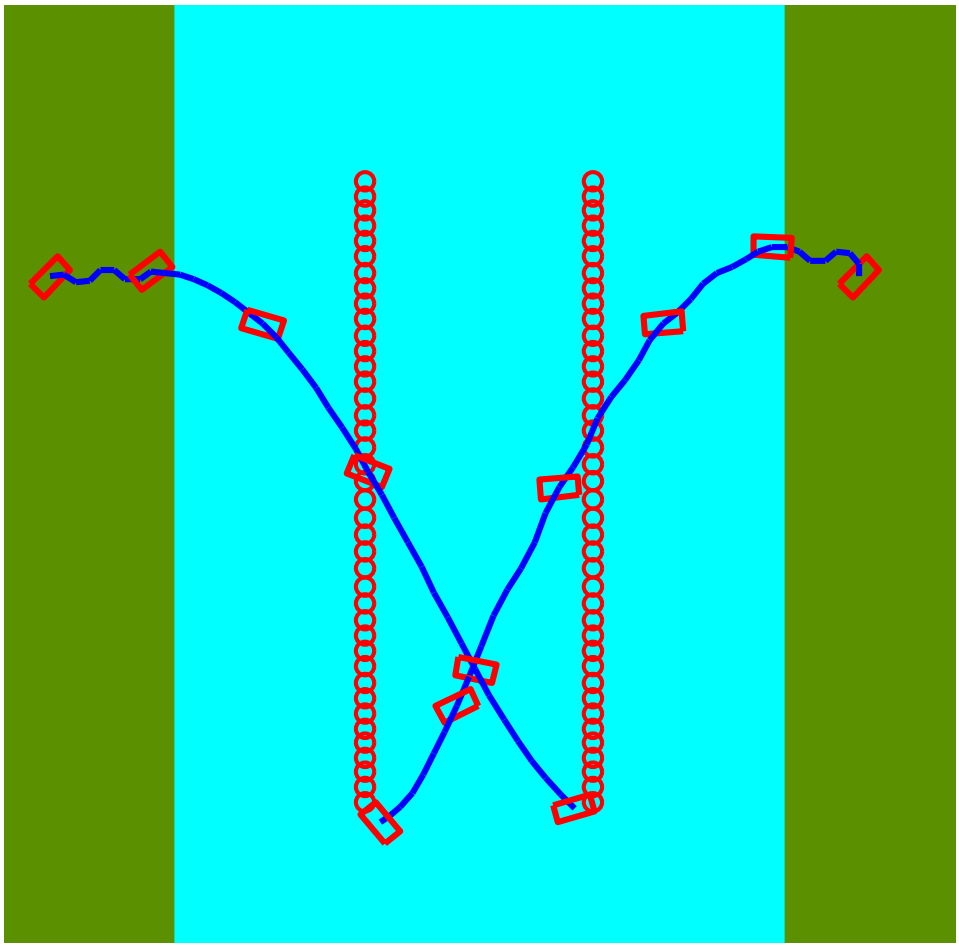


Figure 6.4: Simulation of Scenario I with greedy approach, *average rescue time*=40.5 steps

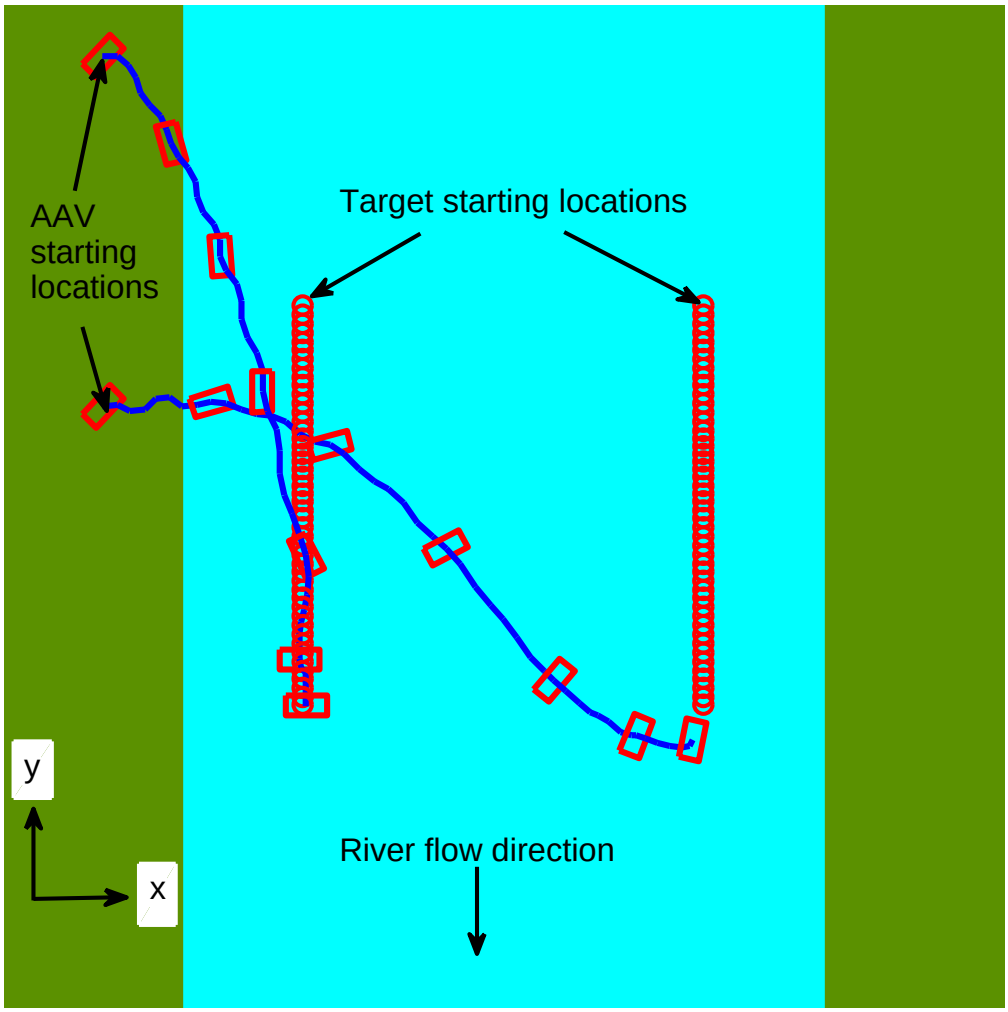


Figure 6.5: Simulation of Scenario II with NBO approach, *average rescue time*=42 steps

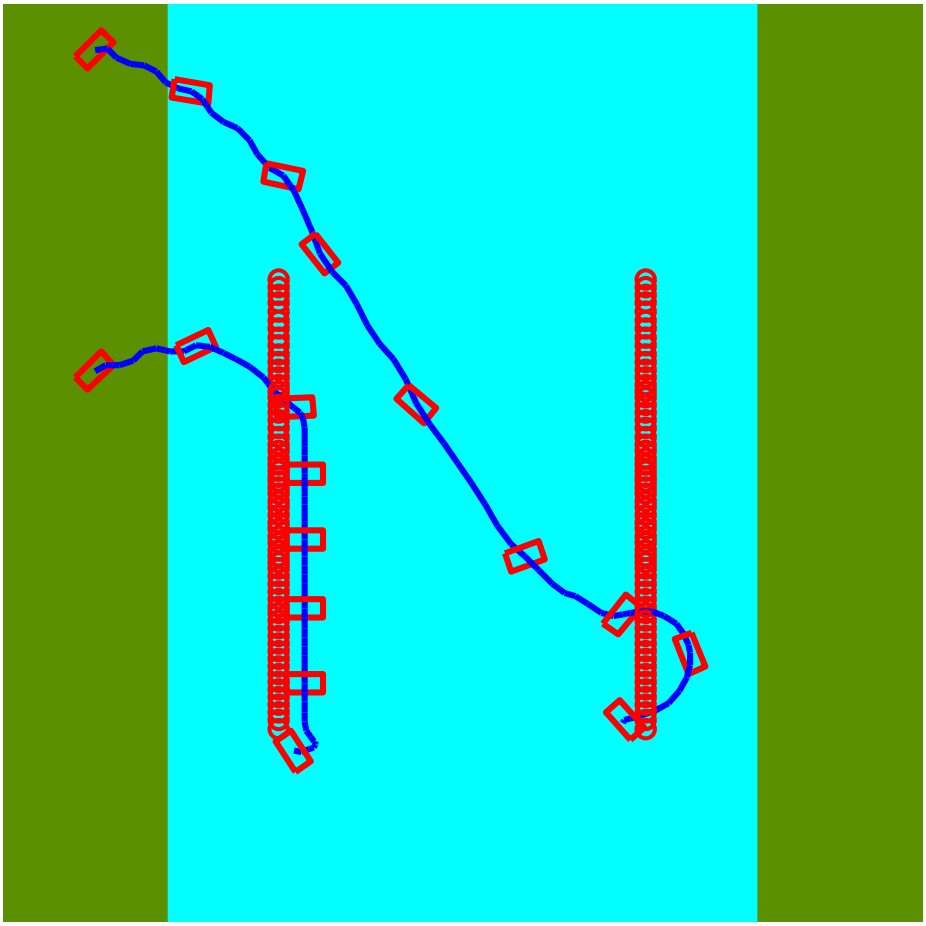


Figure 6.6: Simulation of Scenario II with greedy approach, *average rescue time*=62.5 steps

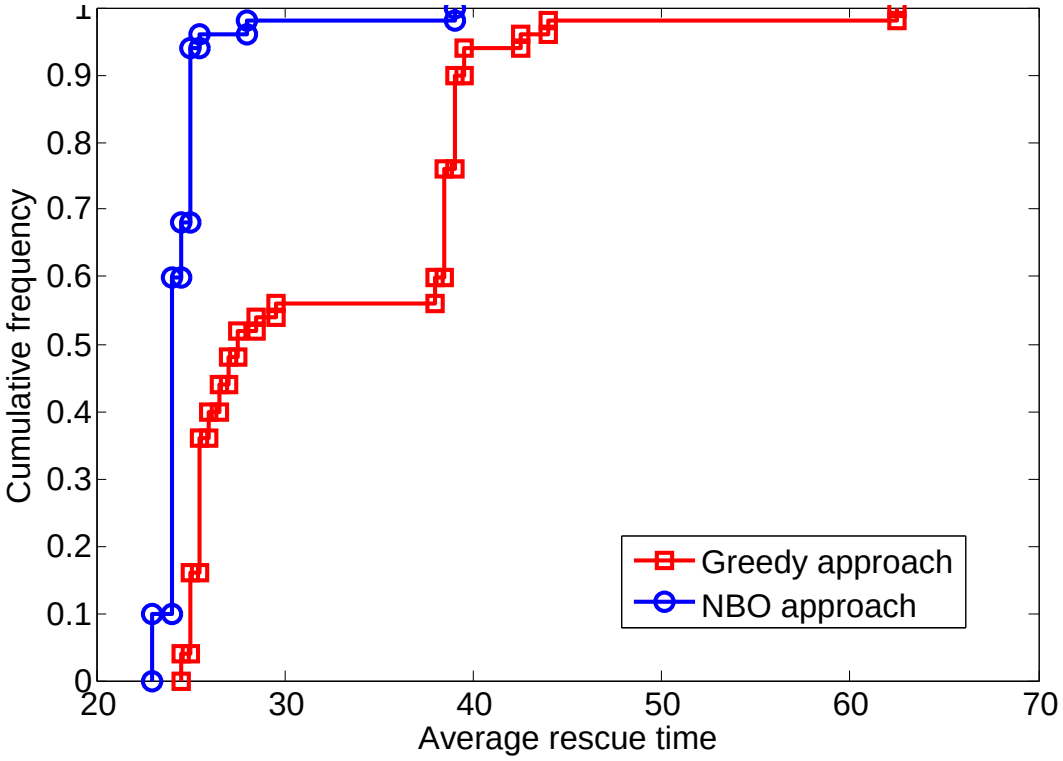


Figure 6.7: Performance comparison for Scenario I: NBO approach vs. greedy approach

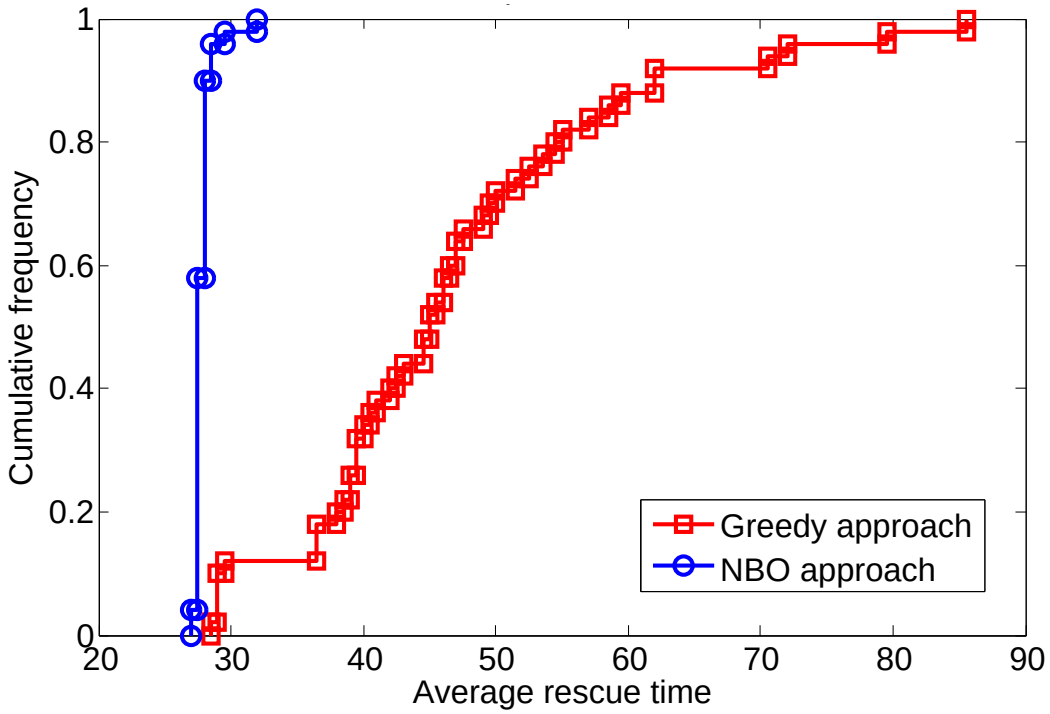


Figure 6.8: Performance comparison for Scenario II: NBO approach vs. greedy approach

CHAPTER 7

DIRECTIONAL SENSOR CONTROL: HEURISTIC APPROACHES

7.1 Introduction

Directional sensors constitute a class of sensors that have a limited field-of-view, e.g., surveillance cameras, infrared sensors, and ultrasound sensors. The methods to control the directions of these sensors are gaining importance owing to a wide range of applications, including surveillance, detection (e.g., human feature detection), and tracking. In this study, we develop tractable solutions to the problem of assigning directions to multiple 2-D directional sensors to maximize the information gain corresponding to multiple target locations. Directional sensor control has been studied before in various contexts [73–76]; a general survey of this topic can be found in [77], where the focus is on coverage issues in directional sensor networks.

In this study, we assume a joint prior Gaussian distribution for the target locations, and we assume that the sensor locations are known exactly. A directional sensor has a limited field-of-view (FOV), where the area sensed by the sensor is given by a sector in a circular region around the sensor as depicted in Figure 7.1. The direction of the FOV of a sensor can be changed by appropriately changing the direction of the sensor. The direction of a sensor can take several discrete values in the interval $[0, 2\pi)$. A directional sensor generates a measurement of a target if and only if the target lies within the FOV of the sensor. We assume that there is a notional fusion center, which collects the measurements from each sensor, and fuses them to form global estimates of target locations.

The objective is to assign a direction to each sensor while maximizing an objective function based on the information gain corresponding to target locations. This problem is hard to solve exactly because of its combinatorial nature, and also the computation time increases exponentially with the number of sensors. In this study, we develop polynomial-time heuristic approaches, and provide bounds on the optimal information gain. We apply rollout on these heuristic approaches (as in [78]) via a dynamic programming formulation [2] to improve the performance of our heuristics with respect to the above objective function.

We then address the above problem using a different formulation, where the objective is to find a mapping of sensors to targets while maximizing the above-mentioned objective function. This problem is also combinatorial in nature, so we extend one of the heuristic approaches, developed for the previous formulation, to solve the mapping problem approximately.

Section 7.2 provides a detailed description of the problem. In Section 7.3, we discuss various heuristic approaches to solve the above problem approximately, and we discuss natural sufficient conditions on objective functions that lead to provable guaranteed performance for our heuristics. However, we show via counterexamples that our objective function does not satisfy these sufficient conditions, suggesting that the nature of the problem is highly non-trivial. Sections 7.4 and 7.5 provide simulation results and concluding remarks respectively.

7.2 Problem Specification

Targets. There are N targets in 2-D, where $(\chi_1, \chi_2, \ldots, \chi_N)$ represent the locations of the targets. The target locations are not known exactly. However, we assume a joint prior Gaussian distribution for the target locations.

Directional Sensors. There are M directional sensors in 2-D, where $(s_1, s_2, \ldots, s_M)$ represent the locations of the sensors. The sensor locations are known exactly. Let $\Theta = \{1, \ldots, K\}$ be the set of directions each sensor can take, where each direction is a value in the interval $[0, 2\pi)$. Let $u = (u_1, \ldots, u_M)$ be the control vector, where $u_i \in \Theta$, $i = 1, \ldots, M$,

is the direction of the ith sensor. The field-of-view of a sensor, pointed at a particular direction $\theta \in \Theta$, is shown in Figure 7.1, where r and α are the radial and angular sensing ranges of the sensor respectively. Because our focus is on solution methods for the problem of controlling directional sensors, rather than on detailed sensor modeling, we adopt a simple 2-D sensing model as shown in Figure 7.1. The 2-D conical field-of-view model in Figure 7.1 is an appropriate approximation to the sensing behavior of many directional sensors, including surveillance cameras [79] and phased arrays [80]. In the case of surveillance cameras, the limited radial range in our sensing model is well justified by the constraint that the camera cannot detect the presence of targets up to a given maximum size located outside a certain range from the camera, when the size of the target image is smaller than a single pixel.

Measurement Errors. Each sensor generates a 2-D position measurement of a target only if the target lies within the FOV of the sensor. These measurements are corrupted by random errors that depend on the relative location of the target with respect to the sensor and the direction of the sensor. The measurement of the jth target at the ith sensor is given by

$$z_{ij} = \begin{cases} \mathbf{H}\chi_j + n_{ij} & \text{if target lies within} \\ & \text{FOV of sensor,} \\ \text{no measurement} & \text{otherwise,} \end{cases}$$

where $n_{ij} \sim \mathcal{N}(0, \mathbf{Z}(s_i, u_i, \chi_j))$, $\mathbf{H}$ is an observation model, and $\mathbf{Z}(\cdot)$ is the measurement error-covariance matrix, which depends on the direction of the sensor and the locations of the target and the sensor.

Fusion. The observations obtained from the sensors are fused to form a global estimate for each target. Let $\mathcal{N}(\xi_j^{\text{prior}}, \mathbf{P}_j^{\text{prior}})$, $j = 1, \ldots, N$, be the prior distributions (Gaussian) of the target-locations. Given the observations and the prior distributions, we evaluate the posterior distribution of the target-locations by fusing the observations. The target observations are not Gaussian; the evaluation of the true Bayesian posterior distribution is not tractable. Therefore, we approximate the posterior distribution of jth target as $\mathcal{N}(\xi_j, \mathbf{P}_j)$,

$j = 1, \ldots, N$, where ξ_j and $\mathbf{P}_j$ are evaluated according to Algorithm 2, where z_{ij} is the observation generated at sensor i.

Algorithm 2 Approximate Posterior Distribution

$A = \left[\mathbf{P}_j^{\text{prior}}\right]^{-1}$
$y = A\xi_j^{\text{prior}}$
for $i = 0$ to M **do** ▷ Information filtering equations
  if Sensor i generates observation **then**
    $A \leftarrow A + \mathbf{H}^{\mathrm{T}} \left[\mathbf{Z}\left(s_i, u_i, \xi_j^{\text{prior}}\right)\right]^{-1} \mathbf{H}$
    $y \leftarrow y + \mathbf{H}^{\mathrm{T}} \left[\mathbf{Z}\left(s_i, u_i, \xi_j^{\text{prior}}\right)\right]^{-1} z_{ij}$
  end if
end for
$\mathbf{P}_j = A^{-1}$
$\xi_j = A^{-1}y$

Objective. The objective is to compute u, i.e., the directions for the sensors, such that the following objective function (based on information gain) is maximized:

$$-\mathrm{E}\left[\sum_{j=1}^{N} \log\det(\mathbf{P}_j)\right],$$

where the expectation is over the prior joint-distribution of target locations, and $\mathbf{P}_j$ is the posterior distribution of the jth target, which is evaluated using Algorithm 2 given the locations of the targets—these target locations are used only to check if the targets fall within the FOV of the sensors.

Information theory provides a way of quantifying the amount of signal-related information that can be extracted from a measurement. This theory also provides tools for assessing the fundamental limitations of different measurement systems in achieving objectives such as detection and tracking, and these fundamental limits can be related to the amount of information gain associated with a specific measurement method. This motivated us to choose an information-theoretic objective function (see [81] for more arguments on behalf of using an information-theoretic objective function).

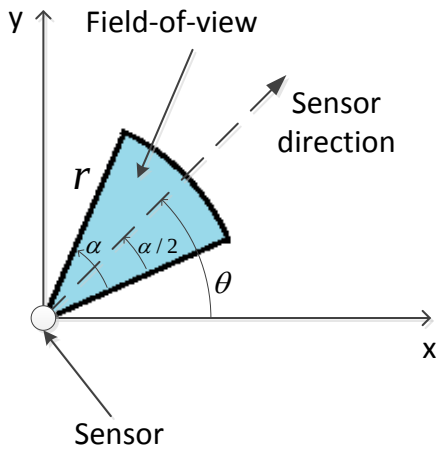


Figure 7.1: Field-of-view of a sensor

Optimal Solution. The optimal directions for the sensors are given by

$$u^* = \arg \max_{u \in \Theta^M} R(u), \tag{7.1}$$

where

$$R(u) = -\mathrm{E}\left[\sum_{j=1}^{N} \log \det(\mathbf{P}_j(u))\right] \tag{7.2}$$

and $\mathbf{P}_j(u)$, $j = 1, \ldots, N$, are evaluated according to Algorithm 2 given the control vector u. We approximate the expectation by a Monte Carlo method. Specifically, we generate several samples from the joint prior distribution of the target locations, and we compute the average (over the samples) objective function value for a given control action. The above problem is a combinatorial optimization problem, where the computational time required to find the optimal solution is $O(K^M)$. Since the computational time increases exponentially with the number of sensors M, we are interested in deriving tractable heuristic methods that are polynomial with respect to the number of sensors.

7.3 Approximate Solutions

7.3.1 Continuous Optimization

We obtain an upper bound on the optimal objective function value by "relaxing" the discrete property of our problem and solving its continuous version. The continuous version

of our combinatorial optimization problem is stated as follows:

$$\begin{aligned} &\underset{(u_1,\ldots,u_M)}{\text{maximize}} \quad R(u_1,\ldots,u_M) \\ &\text{subject to} \quad 0 \leq u_i < 2\pi,\ i = 1,\ldots,M. \end{aligned}$$

The optimal objective function value stands as an upper bound to the optimal objective function value of our original problem. The solution to the above problem can be obtained via a nonlinear programming (NLP) solver. Specifically, we adopt the simulated annealing algorithm to solve the above problem.

In the following subsection, we present several heuristic approaches to solve our problem approximately. The solutions of these heuristics provide a lower bound on the optimal objective function value (discussed later). Because it is hard to compute the optimal objective function value, we use the upper bound above on the optimal objective function value, i.e., the solution from the above-mentioned continuous optimization approach, to see how close the heuristics are to the optimal.

7.3.2 Heuristic Approaches

Let $u = (u_1, u_2, \ldots, u_M)$ represent a solution to our problem. The optimal solution is given by

$$u^* = \arg \max_{u_i \in \Theta, i=1\ldots,M} R(u_1,\ldots,u_M),$$

where $\Theta = \{1,\ldots,K\}$. As we discussed earlier, the computation complexity to obtain the optimal solution through (7.1) is $O(K^M)$, which is exponential in the number of sensors M. So, we are interested in developing heuristic approaches that are polynomial in the number of sensors. The following algorithm, called $\mathcal{H}_1$, generates the solution $u_{\mathcal{H}_1} = (\bar{u}_1,\ldots,\bar{u}_M)$,

where

$$\begin{aligned}
\bar{u}_1 &= \arg\max_{u\in\Theta} R(u, \emptyset, \ldots, \emptyset), \\
&\vdots \\
\bar{u}_k &= \arg\max_{u\in\Theta} R(\bar{u}_1, \ldots, \bar{u}_{k-1}, u, \emptyset, \ldots, \emptyset), \\
&\vdots \\
\bar{u}_M &= \arg\max_{u\in\Theta} R(\bar{u}_1, \ldots, \bar{u}_{M-1}, u),
\end{aligned}$$

where $\emptyset$ at any jth location in the solution $u = (u_1, \ldots, u_M)$, i.e., u_j replaced by $\emptyset$, means that the senor j is ignored while computing the objective function (as if the sensor j does not generate any observations and does not influence the objective function). Therefore, the algorithm $\mathcal{H}_1$ generates an approximate solution $u_{\mathcal{H}_1} = (\bar{u}_1, \ldots, \bar{u}_M)$, and the objective function value from $\mathcal{H}_1$ is $R(\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_M)$. The computational complexity of $\mathcal{H}_1$ is $O(KM)$, which is now linear with respect to the number of sensors.

We now present a second heuristic approach $\mathcal{H}_2$, where $\mathcal{H}_2$ generates the solution $u_{\mathcal{H}_2} = (\hat{u}_1, \ldots, \hat{u}_M)$. In contrast to the previous heuristic approach, this approach may not generate the elements of the solution vector in the order $\hat{u}_1, \ldots, \hat{u}_M$. Let $(p_1, \ldots, p_M)$ be the order in which the elements of the solution vector are generated, where $(p_1, \ldots, p_M)$ is a permutation of $(1, \ldots, M)$. This algorithm is evaluated stepwise, with a total of M steps. At each step, a sensor is assigned a direction, and this assignment is fixed for the rest of the steps, as described below.

1. In the first step, for each sensor i, we associate a direction that maximizes the objective function that depends only on the direction of sensor i, i.e., we ignore the rest of the sensors when we are associating a direction to the sensor i. Let p_1 and $\hat{u}_{p_1}$ be the sensor and its associated direction respectively that gives the maximum objective function

value among all the associations. Now, we assign direction $\hat{u}_{p_1}$ to sensor p_1, and this assignment is fixed for the rest of the steps in the algorithm.

2. In the kth step, we associate a direction to each sensor $j \in \{1, \ldots, M\} \setminus \{p_1, \ldots, p_{k-1}\}$, while maximizing the objective function, which now depends on the directions of the sensors $p_1, \ldots, p_{k-1}$ (computed in the previous $k-1$ steps) and the direction of sensor j. Let $(p_k, \hat{u}_{p_k})$ be the sensor-direction pair that has the maximum objective function value from the above computation. At kth step, we assign direction $\hat{u}_{p_k}$ to the sensor p_k.

3. We repeat the above step until the last sensor p_M is assigned a direction.

4. At the end of the algorithm, we are left with the solution $(\hat{u}_1, \ldots, \hat{u}_M)$.

The computational complexity of this approach is $O(KM^2)$. In both the heuristics $\mathcal{H}_1$ and $\mathcal{H}_2$, since we do not search all possible directions exhaustively, the objective function values from these approaches stand as lower bounds on the optimal value.

Note that the algorithm $\mathcal{H}_1$ requires an ordering among the sensors, and the objective function value from $\mathcal{H}_1$ depends on this order. But the objective function value from $\mathcal{H}_2$ is independent of the ordering of sensors as $\mathcal{H}_2$ does not require any ordering among the sensors.

7.3.3 Rollout on a Heuristic Approach

Given a heuristic approach that solves a combinatorial optimization problem step-wise like our $\mathcal{H}_1$, the authors of [78] have utilized dynamic programming formulation [2] to improve the performance of the heuristic. Specifically, they use an approximate dynamic programming approach called rollout to improve the performance of the given heuristic. We adopt a similar technique, and apply rollout on our heuristics to improve their performance (with respect to the objective function value).

We can obtain the exact optimal solution (step-wise) using the dynamic programming [2] approach as follows. We start at a dummy (artificial) initial state; the state of the algorithm at the 1st stage is (u_1). The state (of the algorithm) at the kth stage is of the form $(u_1, \ldots, u_k)$, also called k-solution. The terminal state is $(u_1, \ldots, u_M)$. The control variable at state $(u_1, \ldots, u_{k-1})$ is $u_k \in \Theta$. We get a reward at the end of the Mth step called terminal reward, which is given by our original objective function $R(u_1, \ldots, u_M)$. Let $J^*(u_1, \ldots, u_k)$ be the optimal value-to-go (see [78] for details) starting from the k-solution, which is the optimal terminal reward given that $(u_1, \ldots, u_k)$ are already assigned to the sensors $1, \ldots, k$. The optimal solution to our problem $(u_1^*, u_2^*, \ldots, u_M^*)$ can be obtained from the following equations:

$$u_k^* = \arg\max_{u \in \Theta} J^*(u_1^*, \ldots, u_{k-1}^*, u), \quad k = 1, \ldots, M. \tag{7.3}$$

In general, the optimal value-to-go $J^*(\cdot)$ is hard to obtain, which is in fact true for our problem. For practical purposes, $J^*(\cdot)$ is replaced with a heuristic value-to-go $\bar{J}(\cdot)$, which is usually easy to obtain.

Let $\mathcal{H}$ be any heuristic algorithm, which generates the path of states $(\bar{i}_1, \bar{i}_2, \ldots, \bar{i}_M)$, where $\bar{i}_k = (\bar{u}_1, \ldots, \bar{u}_k)$. Let $\bar{J}(\bar{i}_k)$ represent the heuristic value-to-go starting from the k-solution $\bar{i}_k = (\bar{u}_1, \ldots, \bar{u}_k)$, from the algorithm $\mathcal{H}$, i.e., we use $\mathcal{H}$ to evaluate the value-to-go. The value-to-go from the algorithm $\mathcal{H}$ is equal to the terminal reward obtained from the algorithm $\mathcal{H}$, i.e., $\bar{J}(\bar{i}_k) = R(\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_M)$. Therefore, the following is true: $\bar{J}(\bar{i}_1) = \bar{J}(\bar{i}_2) = \ldots = \bar{J}(\bar{i}_M)$. We use this heuristic value-to-go in (7.3) to find an approximate solution to our problem. We call this approximation algorithm "Rollout on $\mathcal{H}$" ($\mathcal{RH}$ in short; the same notation was used in [78]) due to its structure, which is similar to an approximate dynamic programming approach called *rollout*. The $\mathcal{RH}$ algorithm starts with the original dummy state, and generates the path $(i_1, i_2, \ldots, i_M)$ according to the following equation:

$$i_k = \arg\max_{j \in N(i_{k-1})} \bar{J}(j), \quad k = 1, \ldots, M$$

where, $i_{k-1} = (u_1, \ldots, u_{k-1})$, and

$$N(i_{k-1}) = \{(u_1, \ldots, u_{k-1}, u) | u \in \Theta\}, \quad k = 1, \ldots, M.$$

The following lemma is adapted from [78]. For completeness, we provide its proof.

Lemma 3.1: The algorithm $\mathcal{RH}$ is sequentially improving with respect to $\mathcal{H}$, i.e., $\bar{J}(i_1) \leq \bar{J}(i_2) \leq \ldots \leq \bar{J}(i_M)$, where $(i_1, i_2, \ldots, i_M)$ is the path generated by the $\mathcal{RH}$ algorithm.

Proof: Let $(i_1, i_2^{i_1}, \ldots, i_M^{i_1})$ be the complete solution from $\mathcal{H}$ given i_1, which is obtained from the first step of $\mathcal{RH}$. We can easily verify that $\bar{J}(i_1) = \bar{J}(i_2^{i_1})$. Let i_2 be the solution obtained from the second step of $\mathcal{RH}$, i.e.,

$$\bar{J}(i_2) = \max_{j \in N(i_1)} \bar{J}(j).$$

But

$$\max_{j \in N(i_1)} \bar{J}(j) \geq \bar{J}(i_2^{i_1}) = \bar{J}(i_1).$$

Therefore, $\bar{J}(i_2) \geq \bar{J}(i_1)$. We can extend this argument for the rest of the steps in $\mathcal{RH}$, which proves the result $\bar{J}(i_1) \leq \bar{J}(i_2) \leq \ldots \leq \bar{J}(i_M)$. ■

The authors of [78] have argued that rollout on a heuristic performs no worse than the heuristic in a different context. We will now extend this argument to our problem and prove that $\mathcal{RH}$ outperforms $\mathcal{H}$.

Theorem 3.2: The algorithm $\mathcal{RH}$ outperforms $\mathcal{H}$, i.e., $R(\bar{i}_M) \leq R(i_M)$, where $\bar{i}_M$ and i_M are the final paths generated by the algorithms $\mathcal{H}$ and $\mathcal{RH}$ respectively.

Proof: We can easily verify that $\bar{J}(\bar{i}_1) = \bar{J}(\bar{i}_2) = \ldots = \bar{J}(\bar{i}_M) = R(\bar{i}_M)$. Since $(i_1, \ldots, i_M)$ is the path generated by $\mathcal{RH}$, and since

$$i_1 = \arg\max_{j \in \Theta} \bar{J}(j),$$

the following is true: $\bar{J}(i_1) \geq \bar{J}(j)$ for all $j \in \Theta$. Since $\bar{i}_1 \in \Theta$, therefore $\bar{J}(i_1) \geq \bar{J}(\bar{i}_1)$. Therefore, from the above result and Lemma 3.1, we can obtain the following result:

$$R(i_M) = \bar{J}(i_M) \geq \ldots \geq \bar{J}(i_1) \geq \bar{J}(\bar{i}_1) = R(\bar{i}_M).$$

■

The above result proves that applying rollout on a heuristic approach guarantees to improve the performance of the heuristic with respect to any objective function. The above rollout approach can be viewed as a one-step lookahead approach (or simply one-step rollout), as we optimize, at every stage, the control for the current step by maximizing the value-to-go given the control for the current step. At the expense of increased computational burden, we can further improve the solution from the above rollout by the following approach: optimize the controls for the current and the next steps combined (i.e., for two steps) by maximizing the value-to-go given the controls for the current and the next steps. This can be viewed as a two-step rollout. Similarly, we can generalize this to an m-step rollout; however as m increases, the computational requirement also increases. When $m = M$, the rollout approach finds the exact optimal solution by exhaustively searching through all possible directions, with computational complexity $O(K^M)$ as in the case of (7.1).

7.3.4 Mapping of Sensors to Targets

In this subsection, our problem formulation differs from the formulation in the previous section. Here, our goal is to map the sensors to the targets while maximizing the objective function defined in the previous subsection. Mapping a sensor to a target gives rise to an assignment of a specific direction to the sensor via the following procedure: a direction that minimizes the angular difference between the direction of the sensor and the direction of the mean of the target's a priori distribution with respect to the sensor's location. Essentially, we are again evaluating the directions for sensors, as in the previous subsections, although indirectly via evaluating a mapping from the set of sensors to the set of targets. The motivation behind formulating this mapping problem is that if the number of targets is less than the number of directions a sensor can take, then the set of feasible solutions is smaller for this new formulation of the problem.

The above mapping problem is also a combinatorial optimization problem, where the computational complexity is now $O(N^M)$, where N is the number of targets and M is the

number of sensors. Therefore, we can simply use the heuristic approaches developed in the previous section to solve the current problem, where the set of controls for each sensor now is the set of targets, in contrast to sensor directions being controls in the previous problem. For a given mapping from sensors to targets, we compute the objective function value as described in (7.2) given the directions of sensors, which are indirectly obtained from a given mapping of sensors to targets.

Let us extend the heuristic algorithm $\mathcal{H}_1$ from the previous section to solve the above problem, and let this new heuristic algorithm be called $\mathcal{MH}_1$ (short for mapping heuristic). In these heuristics, as discussed before, we evaluate directions to sensors stage-wise, i.e., assign a direction to the first sensor, then to the second sensor, and so on. Let u_k be the direction assigned to the kth sensor at stage k. Let

$$U = \{(u_1, \ldots, u_k) | k = 1, \ldots, M, u_k \in \Theta\}$$

be the set of all possible stage-wise controls, where $(u_1, \ldots, u_k)$ at kth stage are the assigned directions to sensors $1, \ldots, k$ respectively. We can notice that the objective function in heuristic algorithm $\mathcal{H}_1$ is defined on the set U.

At this point, it would be interesting to ask if we can provide performance guarantees for $\mathcal{H}_1$ over $\mathcal{MH}_1$. Naturally, such a result would necessitate imposing appropriate restrictions on the objective function. In the following, we will explore what seems to be a reasonable such restriction, based on which we will prove that under such a restriction $\mathcal{H}_1$ outperforms $\mathcal{MH}_1$. However, as we will see later, our problem is sufficiently nontrivial as to frustrate even a reasonable sufficient condition.

To proceed with this investigation, we now provide a definition of a reasonable sufficient condition under which a provable performance guarantee can be achieved.

Definition 1: Given any function $R : U \to \mathbb{R}$, and for every pair of elements in U of the form $(\hat{u}_1, \ldots, \hat{u}_k)$ and $(\bar{u}_1 \ldots, \bar{u}_k)$ for any k that satisfies the condition $R((\hat{u}_1, \ldots, \hat{u}_k)) \geq$

$R((\bar{u}_1 \ldots, \bar{u}_k))$, then R is said to be *continuous monotone* if

$$R((\hat{u}_1, \ldots, \hat{u}_k, a)) \geq R((\bar{u}_1 \ldots, \bar{u}_k, a)),$$

for every $a \in \Theta$.

Example Let $r_i : \Theta \to \mathbb{R}$ for $i = 1, \ldots, M$, and let

$$R((u_1, \ldots, u_k)) = \sum_{i=1}^{k} r_i(u_i),$$

for $k = 1, \ldots, M$. We can easily verify that the above *additive* objective function is continuous monotone.

Let $R : U \to \mathbb{R}$ be any generic objective function, and let $u_{\mathcal{H}_1}$ be the directions for sensors obtained from $\mathcal{H}_1$ and let $u_{\mathcal{MH}_1}$ be the directions for sensors from $\mathcal{MH}_1$ using the objective function R.

Theorem 3.3: If R is continuous monotone, the algorithm $\mathcal{H}_1$ outperforms $\mathcal{MH}_1$ with respect to R, i.e., $R(u_{\mathcal{H}_1}) \geq R(u_{\mathcal{MH}_1})$.

Proof: Let $u_{\mathcal{H}_1} = (\hat{u}_1, \ldots, \hat{u}_M)$ and $u_{\mathcal{MH}_1} = (\bar{u}_1, \ldots, \bar{u}_M)$. The first element of $u_{\mathcal{H}_1}$, i.e., $\hat{u}_1$ is evaluated as follows:

$$\hat{u}_1 = \arg\max_{a \in \Theta} R((a)),$$

whereas $\bar{u}_1$ is obtained (albeit indirectly) by assigning sensor 1 to each target and checking which assignment maximizes the objective function value. In other words, there exists $\Theta_1 \subseteq \Theta$ such that

$$\bar{u}_1 = \arg\max_{a \in \Theta_1} R((a)),$$

which implies that $R((\hat{u}_1)) \geq R((\bar{u}_1))$. Since R is continuous monotone, by definition $R((\hat{u}_1, a)) \geq R((\bar{u}_1, a))$ for every $a \in \Theta$. Therefore, $\max_{a \in \Theta} R((\hat{u}_1, a)) \geq \max_{a \in \Theta} R((\bar{u}_1, a))$. We can extend the above discussion on the first step of $\mathcal{MH}_1$ algorithm to its second step, i.e., there exists $\Theta_2 \subseteq \Theta$ such that $\bar{u}_2 = \arg\max_{a \in \Theta_2} R((\bar{u}_1, a))$. We can easily verify that

$\max_{a\in\Theta} R((\hat{u}_1, a)) \geq \max_{a\in\Theta_2} R((\bar{u}_1, a))$ or $R((\hat{u}_1, \hat{u}_2)) \geq R((\bar{u}_1, \bar{u}_2))$. We can extend this argument for the rest of the steps, which proves the result $R((\hat{u}_1, \ldots, \hat{u}_M)) \geq R((\bar{u}_1, \ldots, \bar{u}_M))$.

■

Unfortunately, as alluded to before, things do not turn out as conveniently as one would have desired. More specifically, our objective function (7.2) is in fact *not* continuous monotone. We will demonstrate this claim with the following counterexample. Nonetheless, the above theorem provides a useful result comparing the performance of $\mathcal{H}_1$ and $\mathcal{MH}_1$ for any generic continuous monotone objective function.

Example Let us consider a scenario with two sensors $\mathbf{a}, \mathbf{b}$ and two targets as shown in Figures 7.2 and 7.3. In these figures, the FOV of each sensor is $\pi/5$, i.e., $\alpha = \pi/5$ in Figure 7.1. We approximate the expectation in the objective function by the following Monte Carlo method. We generate 50 samples from the (joint) target location distribution. For a given control vector u, we compute the objective function $R(u)$ from each sample; the objective function value for the given control vector is given by the average of these 50 objective function values. In these figures, the sensors are represented by small circles, the target (prior) distributions are represented by the error concentration ellipses, and the FOVs are represented by 2-D cones. For this scenario, we compute our objective function values for the following two cases:

1. sensors **a** and **b** are assigned directions 0 and $\pi/2$ respectively as shown in Figure 7.2, and

2. sensors **a** and **b** are assigned directions $-\pi/2$ and $\pi/2$ respectively as shown in Figure 7.3.

The following are the objective function values for the above cases: $R((0)) = 8.05$, $R((-\pi/2)) = 7.36$, $R((0, \pi/2)) = 9.65$, $R((-\pi/2, \pi/2)) = 10.37$. Therefore, $R((0)) > R((-\pi/2))$, but $R((0, \pi/2)) < R((-\pi/2, \pi/2))$, which proves that our objective function (7.2) is not continuous monotone.

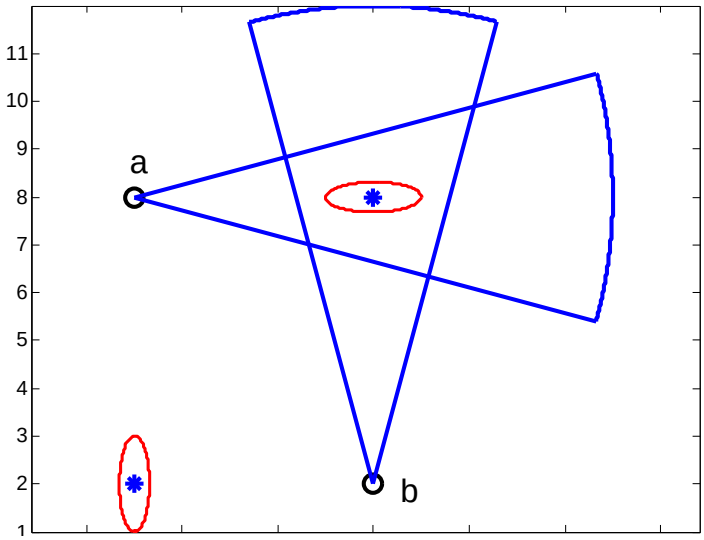


Figure 7.2: Counterexample to show that our objective function is not continuous monotone (Case 1).

7.4 Simulation Results and Further Discussion

We implement the heuristic approaches presented in the previous section in MATLAB for a scenario with six sensors and nine targets, where each sensor can take 10 possible directions $\{0, 2\pi/10, 2(2\pi/10), \ldots, 9(2\pi/10)\}$. Figures 7.4, 7.5, and 7.6 depict the locations of targets, sensors, and the solution from the approaches $\mathcal{H}_1$, $\mathcal{RH}_1$, and $\mathcal{MH}_1$ respectively. The solution from each of these approaches are the directions computed for the sensors (can be interpreted from the FOVs of sensors shown in these figures while using Figure 7.1 as reference). Table 7.1 compares the objective function values of the solutions obtained from each of the heuristic approaches discussed in this study along with the objective function value from continuous (relaxed) optimization. This table corroborates the result in Theorem 3.2 that the rollout on a heuristic approach outperforms the heuristic approach.

Table 7.1 demonstrates that the objective function values from the heuristics are relatively close to that of the relaxed problem (upper bound), which implies that the solutions from our heuristics are close to optimal. This prompts us to question if in fact we could have known beforehand that our heuristics are provably close to optimal because our problem

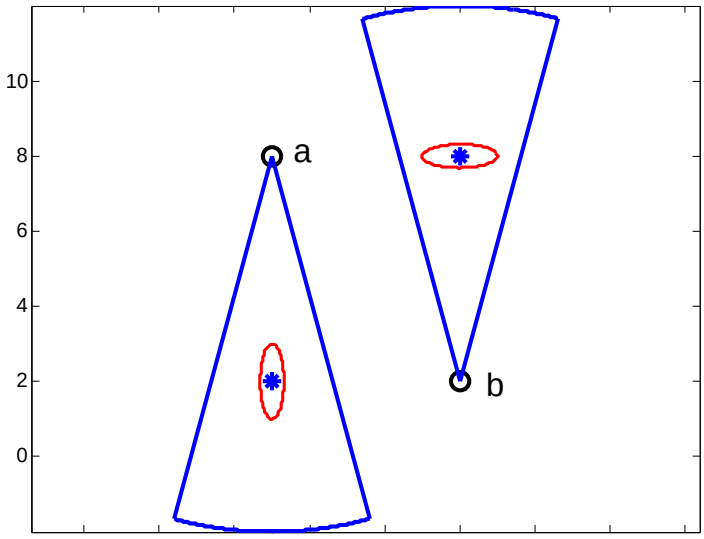


Figure 7.3: Counterexample to show that our objective function is not continuous monotone (Case 2).

Table 7.1: Objective function values from various approaches

Approach	**Objective function value**
Relaxed (upper bound)	46.67
$\mathcal{H}_1$	42.71
$\mathcal{RH}_1$	44.03
$\mathcal{H}_2$	43.76
$\mathcal{RH}_2$	44.93
$\mathcal{MH}_1$	41.06

exhibits properties that are known to result in provable suboptimality bounds. We will now investigate this question.

In a recent study [82,83], it was shown that if an objective function is *string-submodular*[1], then the "greedy strategy" performs at least as good as $(1-1/e) \approx 0.63$ of the optimal. The definition of the above-mentioned greedy strategy in [82] is exactly the same as our heuristic algorithm $\mathcal{H}_1$. The following is an interesting observation from the results in our study. The objective function value from $\mathcal{H}_1$ is

$$R(u_{\mathcal{H}_1}) = 0.91 R_{\text{relaxed}} \geq 0.91 R_{\text{optimal}} \geq (1-1/e) R_{\text{optimal}},$$

[1] An objective function is said to be string-submodular if it has the following properties: *forward-monotone* and *diminishing returns*. See [82] or [83] for the definition of these properties.

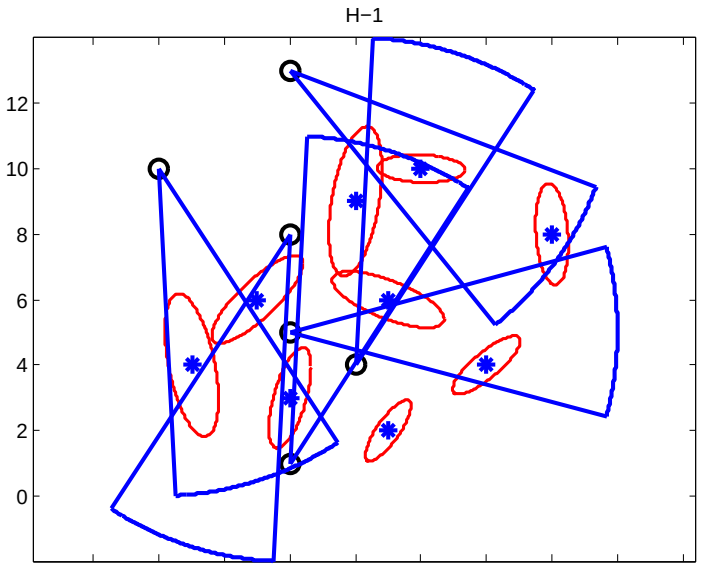


Figure 7.4: Solution from $\mathcal{H}_1$

i.e., the objective function value from algorithm $\mathcal{H}_1$ is at least as good as $(1 - 1/e)R_{\text{optimal}}$. We further compare the objective function values from $\mathcal{H}_1$ and the relaxed approach (which provides an upper bound on the optimal objective function value) for various scenarios with varying numbers of sensors and targets; Table 7.2 summarizes the results.

Table 7.2: Comparison of objective function values from $\mathcal{H}_1$ and the relaxed approach.

Scenario	$R(u_{\mathcal{H}_1})/R_{\text{relaxed}}$
3 sensors, 5 targets	0.92
3 sensors, 9 targets	0.90
4 sensors, 5 targets	0.77
4 sensors, 9 targets	0.92
6 sensors, 9 targets	0.91

Table 7.2 demonstrates that our heuristic algorithm $\mathcal{H}_1$ is at least as good as $(1 - 1/e)R_{\text{optimal}}$ for every scenario considered. This observation suggests that our objective function may have a string-submodular type property. Unfortunately, things are once again not quite as straightforward as one would have expected. Alas, our objective function turns out *not* to be string-submodular, as we show in the following counterexample. This suggests

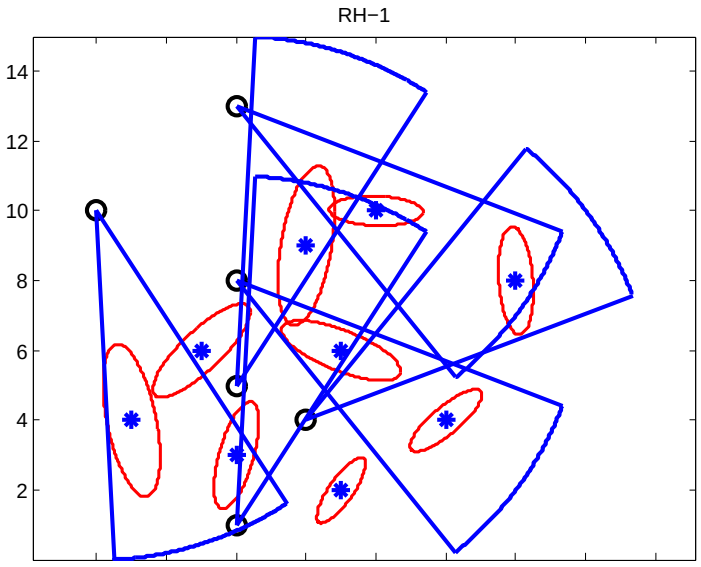


Figure 7.5: Solution from $\mathcal{RH}_1$

that our problem possesses highly nontrivial features that elude currently known analytical machinery.

Example Let us consider a scenario with three sensors and one target as shown in Figure 7.7. Let the sensors be a, b, and c. Without loss of generality, let $a \rightarrow b \rightarrow c$ be the sequence in which directions are computed. Let us assign $\pi/3, 3\pi/4, 3\pi/4$ to the sensors a, b, and c respectively. We can easily verify the following:

$$R((\pi/3,3\pi/4)) - R((\pi/3)) \leq$$
$$R((\pi/3, 3\pi/4, 3\pi/4)) - R((\pi/3, 3\pi/4)),$$

where R is our objective function. The above inequality shows that our objective function does not have the diminishing returns property (again, see [82] or [83] for details), thus proving that our objective function is not string-submodular.

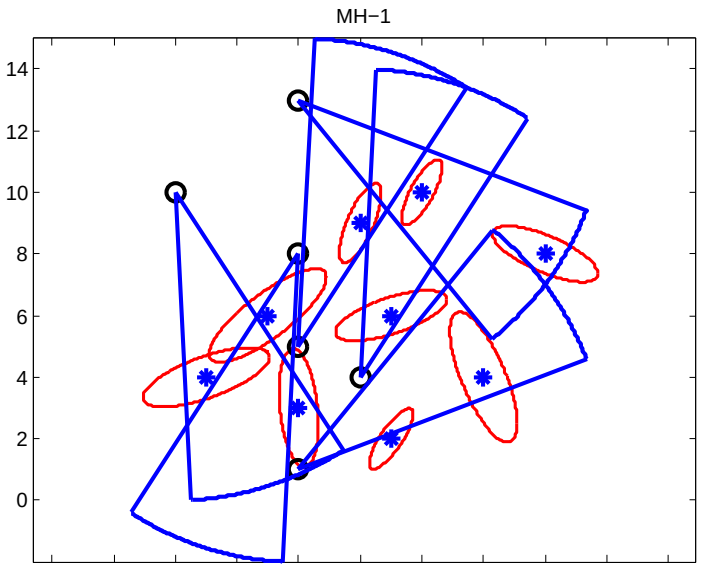


Figure 7.6: Solution from $\mathcal{MH}_1$

7.5 Concluding Remarks

We investigated the problem of controlling the directional sensors for maximizing an information-gain-based objective function. We identified that this problem is a combinatorial optimization problem, and developed heuristic approaches ($\mathcal{H}_1$ and $\mathcal{H}_2$) to solve the problem approximately. We further improved the performance of our heuristics by applying an approximate dynamic programming approach called rollout. The rollout on our heuristic approach outperforms the heuristic approach, and our empirical results are in agreement with this.

We then addressed this problem via a different formulation, where the goal was to find an optimal mapping from the set of sensors to the set of targets that maximizes our information-gain-based objective function. This problem is also combinatorial in nature, so we extended the heuristic approach $\mathcal{H}_1$, developed for the previous formulation, to solve this mapping problem approximately, and we called this new heuristic algorithm $\mathcal{MH}_1$. We investigated natural sufficient conditions on objective functions that lead to provable guaranteed performance for our heuristics. Specifically, we proved that if an objective function is continuous

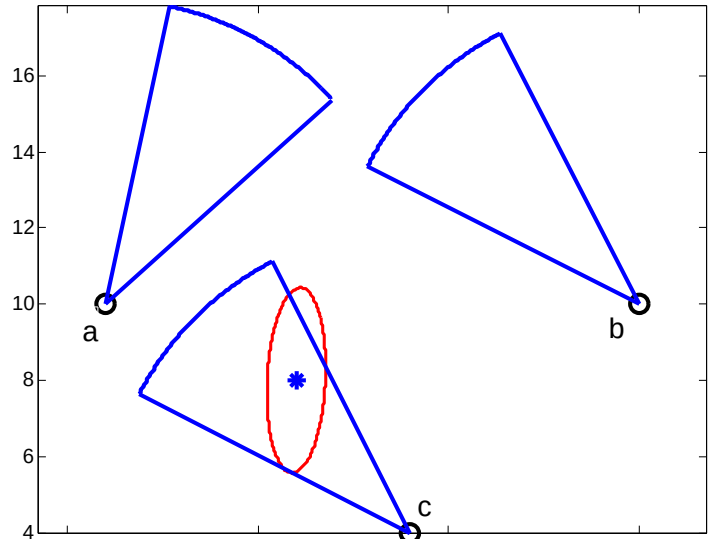


Figure 7.7: Counterexample to show that our objective function is not string-submodular

monotone, then $\mathcal{MH}_1$ outperforms $\mathcal{H}_1$ with respect to the objective function value. However, we showed via a counterexample that our objective function does not satisfy this sufficient condition, suggesting that the nature of the problem is highly nontrivial. Nonetheless, the above result provides an analytical comparison of performances of $\mathcal{H}_1$ and $\mathcal{MH}_1$ for any generic continuous monotone objective function.

Our empirical results show that our heuristic algorithm $\mathcal{H}_1$ performed very close to the upper bound on the optimal, i.e., close to optimal, leading us to wonder whether our objective function possesses a property that would guarantee $\mathcal{H}_1$ to perform close to optimal. So, we investigated this question, and found a recent study in the literature that showed that if an objective function is string-submodular, then the greedy strategy ($\mathcal{H}_1$ in our study) performs at least as good as $(1 - 1/e)$ of the optimal. So, we further compared the performance of $\mathcal{H}_1$ for several scenarios with varying numbers of sensors and targets, and the results demonstrate that for each scenario the objective function value from $\mathcal{H}_1$ is at least as good as $(1 - 1/e)$ of the optimal. However, we found a counterexample proving our objective function is in fact not string-submodular. This again suggests that our problem possesses highly nontrivial features that elude currently known analytical machinery. It remains an interesting future

study to understand why the performance of our heuristic algorithm is so close to optimal. The results in this chapter were published in [84, 85].

CHAPTER 8

CONCLUSIONS AND REMARKS

In the first phase of this study, we developed guidance control methods for unmanned aerial vehicles (UAVs) for multitarget tracking in both centralized and decentralized settings. In the centralized setting, the algorithm development was based on the theory of partially observable Markov decision process (POMDP). Because it is hard to solve a POMDP exactly, we adopted an approximation method called nominal belief-state optimization (NBO). We then derived bounds on the optimal cost function value for this problem, and demonstrated through empirical study that the cost function value from NBO is close to the optimal cost's lower bound, thus proving that the NBO performs close to optimal. We then extended the algorithm to incorporate wind disturbance on UAVs, collision avoidance (among UAVs, and between UAVs and obstacles), threat evasion, evasive target tracking, and track-swap avoidance. In the decentralized setting, the algorithm development was based on the theory of decentralized POMDP (Dec-POMDP). A Dec-POMDP is also hard to be solved exactly. Therefore, we extended the above-mentioned NBO method to solve the decentralized UAV guidance problem posed as Dec-POMDP. We then compared the performance of our Dec-POMDP approach with a "greedy" approach. We showed quantitatively how much our Dec-POMDP approach outperforms (with respect to *average target-location error*) the greedy approach.

In the second phase, we developed guidance control methods for autonomous amphibious vehicles (AAVs) for flood rescue support. More precisely, we designed a guidance algorithm for AAVs to rescue human targets stranded in a flood situation. We designed this algorithm based on the theory of POMDPs, and again used the NBO method to solve the POMDP

approximately. Our results show that the NBO approach outperforms a "greedy" approach significantly.

In the third phase, we studied the problem of controlling directional sensors (e.g., surveillance cameras) for maximizing an information-gain-based objective function. More precisely, the goal was to assign a direction (from a discrete and finite set of directions) to each sensor while maximizing the objective function. This turned out to be a combinatorial optimization problem, which is hard to be solved exactly. Therefore, we developed heuristic approaches ($\mathcal{H}_1$ and $\mathcal{H}_2$) to solve the problem approximately. We further improved the performance of our heuristics by applying an approximate dynamic programming approach called rollout. We analytically proved that the rollout on our heuristic approach outperforms the heuristic approach, and corroborated this with empirical results. We then addressed this problem via a different formulation, where the goal was to find an optimal mapping from the set of sensors to the set of targets that maximizes our information-gain-based objective function. This problem is also combinatorial in nature, so we extended the heuristic approach $\mathcal{H}_1$, developed for the previous formulation, to solve this mapping problem approximately, and we called this new heuristic algorithm $\mathcal{MH}_1$. We investigated natural sufficient conditions on objective functions that lead to provable guaranteed performances for our heuristics. Specifically, we proved that if an objective function is continuous monotone, then $\mathcal{MH}_1$ outperforms $\mathcal{H}_1$ with respect to the objective function value. However, we showed via a counterexample that our objective function does not satisfy this sufficient condition, suggesting that the nature of the problem is highly nontrivial. Our empirical results show that our heuristic algorithm $\mathcal{H}_1$ performed very close to the upper bound on the optimal, i.e., close to optimal, leading us to wonder whether our objective function possesses a property that would guarantee $\mathcal{H}_1$ to perform close to optimal. So, we investigated this question, and found a recent study in the literature that showed that if an objective function is "string-submodular", then the $\mathcal{H}_1$ performs at least as good as $(1 - 1/e)$ of the optimal. So, we further compared the performance of $\mathcal{H}_1$ for several scenarios with varying numbers of sensors and targets, and the

results demonstrate that for each scenario the objective function value from $\mathcal{H}_1$ is at least as good as $(1-1/e)$ of the optimal. However, we found a counterexample proving our objective function is in fact not string-submodular. This again suggests that our problem possesses highly nontrivial features that elude currently known analytical machinery. It remains an interesting future study to understand why the performance of our heuristic algorithm is so close to optimal.

REFERENCES

[1] E. K. P. Chong, C. Kreucher, and A. O. Hero, "Partially observable Markov decision process approximations for adaptive sensing," *Disc. Event Dyn. Sys.*, vol. 19, pp. 377–422, 2009.

[2] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[3] C. Kreucher, A. O. Hero, K. Kastella, and D. Chang, "Efficient methods of non-myopic sensor management for multitarget tracking," in *Proc. 43rd IEEE Conf. Decision and Control*, Paradise Island, Bahamas, 2004, pp. 722–727.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[6] D. P. Bertsekas and D. A. Castanon, "Rollout algorithms for stochastic scheduling problems," *J. Heuristics*, vol. 5, pp. 89–108, 1999.

[7] E. K. P. Chong, R. L. Givan, and H. S. Chang, "A framework for simulation-based network control via hindsight optimization," in *Proc. 39th IEEE Conf. Decision and Control*, Sydney, Australia, 2000, pp. 1433–1438.

[8] G. Wu, E. K. P. Chong, and R. Givan, "Burst-level congestion control using hindsight optimization," *IEEE Trans. Autom. Control*, vol. 47, pp. 979–991, 2002.

[9] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2007, vol. 2.

[10] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Oper. Res.*, vol. 27, pp. 819–840, 2002.

[11] S. A. Miller, Z. A. Harris, and E. K. P. Chong, "A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking," *EURASIP J. Adv. Signal Process.*, vol. 2009, 2009.

[12] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006.

[13] C. Geyer, "Active target search from UAVs in urban environments," in *Proc. IEEE Int. Conf. Robotics and Automation*, Pasadena, CA, 2008, pp. 2366–2371.

[14] J. Tisdale, H. Durrant-Whyte, and J. K. Hedrick, "Path planning for cooperative sensing using unmanned vehicles," in *Proc. ASME Int. Mechanical Engineering Conf. and Exposition*, Seattle, WA, 2007, pp. 715–723.

[15] R. He, A. Bachrach, and N. Roy, "Efficient planning under uncertainty for a target-tracking micro-aerial vehicle," in *Proc. IEEE Int. Conf. Robotics and Automation*, Anchorage, AK, 2010.

[16] Y. Kim, D. Gub, and I. Postlethwaite, "Real-time path planning with limited information for autonomous unmanned air vehicles," *Automatica*, vol. 44, pp. 696–712, 2008.

[17] J. B. Saunders, O. Call, A. Curtis, R. W. Beard, and T. W. McLain, "Static and dynamic obstacle avoidance in miniature air vehicles," in *Proc. Infotech at Aerospace Conf.*, Arlington, VA, 2005.

[18] C. G. Cassandras and W. Li, "A receding horizon approach for dynamic UAV mission management," in *Proc. SPIE 17th Annu. Int. Symp.*, Orlando, FL, 2003, pp. 284–293.

[19] X. Ma and D. A. Castanon, "Receding horizon planning for dubins traveling salesman problems," in *Proc. 45th IEEE Conf. Decision and Control*, San Diego, CA, 2006, pp. 5453–5458.

[20] Y. Lu, X. Huo, O. Arslan, and P. Tsiotras, "Incremental multi-scale search algorithm for dynamic path planning with low worst-case complexity," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 41, pp. 1556–1570, 2011.

[21] W. Ren, J. Sun, R. W. Beard, and T. W. McLain, "Nonlinear tracking control for nonholonomic mobile robots with input constraints: an experimental study," in *Proc. American Control Conf.*, Portland, OR, 2005, pp. 4923–4928.

[22] W. Li and C. G. Cassandras, "A cooperative receding horizon controller for multivehicle uncertain environments," *IEEE Trans. Autom. Control*, vol. 51, pp. 242–257, 2006.

[23] P. W. Sarunic, R. J. Evans, and B. Moran, "Control of unmanned aerial vehicles for passive detection and tracking of multiple emitters," in *Proc. IEEE Symp. Computational Intelligence in Security and Defense Applications*, Ottawa, Canada, 2009.

[24] S. Candido and S. Hutchinson, "Minimum uncertainty robot path planning using a POMDP approach," in *Proc. IEEE Int. Conf. Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 1408–1413.

[25] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee, "Motion planning under uncertainty for robotic tasks with long time horizons," *Int. J. Robot. Res.*, vol. 30, pp. 308–323, 2011.

[26] N. Ceccarelli, J. J. Enright, E. Frazzoli, S. J. Rasmussen, and C. J. Schumacher, "Micro UAV path planning for reconnaissance in wind," in *Proc. American Control Conf.*, New York City, NY, 2007, pp. 5310–5315.

[27] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," in *Proc. American Control Conf.*, Minneapolis, MN, 2006, pp. 4261–4266.

[28] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, pp. 566–580, 1996.

[29] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Trans. Syst. Man Cybern.*, vol. 19, pp. 1179–1187, 1989.

[30] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for UAV navigation," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 33, pp. 898–912, 2003.

[31] B. A. Kumar and D. Ghose, "Radar-assisted collision avoidance/guidance strategy for planar flight," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, pp. 77–90, 2001.

[32] S. Temizer, M. J. Kochenderfer, L. P. Kaelbling, T. Lozano-Perez, and J. K. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, Toronto, Canada, 2010.

[33] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Boston, MA: Artech House, 1999.

[34] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. NY: Wiley-Interscience, 2001.

[35] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. London, UK: Academic Press Inc., 1988.

[36] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from adp to mpc," *Fundam. Issues Control, Eur. J. Control*, vol. 11, 2005.

[37] B. R. Geiger, J. F. Horn, A. M. DeLullo, and L. N. Long, "Optimal path planning of UAVs using direct collocation with nonlinear programming," in *Proc. AIAA Guidance, Navigation, and Control Conf.*, Keystone, CO, 2006, paper 2006–6199.

[38] P. Ailliot, V. Monbet, and M. Prevosto, "An autoregressive model with time-varying coefficients for wind fields," *Environmetrics*, vol. 17, pp. 107–117, 2006.

[39] A. V. Boukhanovsky, H. E. Krogstad, L. J. Lopatoukhin, and V. A. Rozhkov, "Stochastic simulation of inhomogeneous metocean fields: part i: annual variability," in *Proc. 2003 Int. Conf. Computational Science*, Melbourne, Australia, 2003, pp. 213–222.

[40] A. Cho, J. Kim, S. Lee, and C. Kee, "Wind estimation and airspeed calibration using a UAV with a single-antenna GPS receiver and pitot tube," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 47, pp. 109–117, 2011.

[41] P. C. Mahalanobis, "On the generalised distance in statistics," in *Proc. Nat. Institute of Sciences of India*, Calcutta, India, 1936, pp. 49–55.

[42] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Ann. Mathematical Statistics*, vol. 22, pp. 79–86, 1951.

[43] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[44] L. L. Cam and G. L. Yang, *Asymptotics in Statistics*. Berlin, Germany: Springer, 2000.

[45] S. Ragi and E. K. P. Chong, "Dynamic UAV path planning for multitarget tracking," in *Proc. American Control Conf.*, Montreal, Canada, 2012, pp. 3845–3850.

[46] ——, "UAV path planning in a dynamic environment via partially observable Markov decision process," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, pp. 2397–2412, 2013.

[47] H. Min, F. Sun, and F. Niu, "Decentralized UAV formation tracking flight control using gyroscopic force," in *Proc. Int. Conf. Computational Intelligence for Measurement Systems and Applications*, Hong Kong, 2009, pp. 91–96.

[48] H. Rezaee and F. Abdollahi, "A synchronization strategy for three dimensional decentralized formation control of unmanned aircrafts," in *Proc. 37th Annu. Conf. IEEE Industrial Electronics Society*, Melbourne, Australia, 2011, pp. 462–467.

[49] Y. Yang, A. A. Minai, and M. M. Polycarpou, "Decentralized cooperative search by networked UAVs in an uncertain environment," in *Proc. American Control Conf.*, Boston, MA, 2004, pp. 5558–5563.

[50] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *Proc. 43rd IEEE Conf. Decision and Control*, Paradise Island, Bahamas, 2004, pp. 4286–4291.

[51] M. Alighanbari and J. P. How, "Decentralized task assignment for unmanned aerial vehicles," in *Proc. 44th IEEE Conf. Decision and Control*, Seville, Spain, 2005, pp. 5668–5673.

[52] A. Ghaffarkhah and Y. Mostofi, "Communication-aware motion planning in mobile networks," *IEEE Trans. Autom. Control*, vol. 56, pp. 2478–2485, 2011.

[53] S. Ragi and E. K. P. Chong, "Decentralized control of unmanned aerial vehicles for multitarget tracking," in *Proc. 2013 Int. Conf. Unmanned Aircraft Systems*, Atlanta, GA, 2013, pp. 260–268.

[54] ——, "Decentralized guidance control of UAVs with explicit optimization of communication," J. Intell. Robot. Syst., to be published.

[55] M. Frejek and S. Nokleby, "Design of a small-scale autonomous amphibious vehicle," in *Proc. Canadian Conf. Electrical and Computer Engineering*, Niagara Falls, Canada, 2008, pp. 781–786.

[56] E. Papadopoulos and M. Misailidis, "On differential drive robot odometry with application to path planning," in *Proc. European Control Conf.*, Kos, Greece, 2007, pp. 5492–5499.

[57] Y. Tee, Y. Tan, B. Teoh, E. Tan, and Z. Wong, "A compact design of zero-radius steering autonomous amphibious vehicle with direct differential directional drive - UTAR-AAV," in *Proc. IEEE Conf. Robotics, Automation, and Mechatronics*, Singapore, 2010, pp. 176–181.

[58] Q. P. Ha, T. H. Tran, S. Scheding, G. Dissanayake, and H. F. Durrant-Whyte, "Control issues of an autonomous vehicle," in *Proc. 22nd Int. Symp. Automation and Robotics in Construction*, Ferrara, Italy, 2005.

[59] W. Masayoshi, "Research and development of electric vehicles for clean transportation," *J. Environ. Sci.*, vol. 21, pp. 745–749, 2009.

[60] T. Brunl, *Embedded Robotics*, 3rd ed. Germany: Springer, 2008.

[61] T. H. Tran, Q. P. Ha, R. Grover, and S. Scheding, "Modelling of an autonomous amphibious vehicle," in *Proc. Australasian Conf. Robotics and Automation*, Canberra, Australia, 2004.

[62] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Autonomous Robots*, vol. 18, pp. 81–102, 2004.

[63] S. Lacroix, A. Mallet, D. Bonnafous, G. Bauzil, S. Fleury, M. Herrb, and R. Chatila, "Autonomous rover navigation on unknown terrains: Functions and integration," *Int. J. Robot. Res.*, vol. 21, 2002.

[64] T. H. Tran, "Modelling and control of unmanned ground vehicles," Ph.D. Dissertation, University of Technology, Sydney, Australia, 2007.

[65] S. A. Watson and P. N. Green, "Design considerations for micro-autonomous underwater vehicles (AUVs)," in *Proc. IEEE Conf. Robotics, Automation, and Mechatronics*, Singapore, 2010, pp. 429–434.

[66] ——, "Propulsion systems for micro-autonomous underwater vehicles (AUVs)," in *Proc. IEEE Conf. Robotics, Automation, and Mechatronics*, Singapore, 2010, pp. 435–440.

[67] L. D. Landau and E. M. Lifshitz, *Fluid Mechanics*, 2nd ed. Pergamon Press, 2000, ch. IV.

[68] C. Kreucher, A. O. H. III, K. Kastella, and D. Chang, "Efficient methods of non-myopic sensor management for multitarget tracking," in *Proc. 43rd IEEE Conf. Decision and Control*, Paradise Island, Bahamas, 2004, pp. 722–727.

[69] E. K. P. Chong, R. L. Givan, and H. S. Chang, "A framework for simulation-based network control via hindsight optimization," in *Proc. 39th IEEE Conf. Decision and Control*, Sydney, Australia, 2000, pp. 1433–1438.

[70] Y. Tee, B. Teoh, D. E. B. Tan, Z. Wong, C. Tan, and Y. Tan, "Design considerations of autonomous amphibious vehicle (utar-aav)," in *Proc. IEEE Conf. Sustainable Utilization and Development in Engineering and Technology*, Petaling Jaya, Malaysia, 2010, pp. 13–18.

[71] S. Ragi, C. S. Tan, and E. K. P. Chong, "Feasibility study of POMDP in autonomous amphibious vehicle guidance," in *Proc. 2013 IFAC Symp. Intelligent Autonomous Vehicles*, Gold Coast, Australia, 2013, pp. 85–90.

[72] ——, "Guidance of autonomous amphibious vehicles for flood rescue support," *Math. Probl. Eng.,*, vol. 2013, 2013.

[73] Y. Wang and G. Cao, "Minimizing service delay in directional sensor networks," in *Proc. 2011 IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 1790–1798.

[74] H. D. Ma and Y. H. Liu, "Some problems of directional sensor networks," *Int. J. Sensor Netorks*, vol. 2, pp. 44–52, 2007.

[75] J. Ai and A. A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *J. Comb. Optim.*, vol. 11, pp. 21–41, 2006.

[76] G. Fusco and H. Gupta, "Placement and orientation of rotating directional sensors," in *Proc. 7th Annu. IEEE Communications Society Conf. SECON*, Boston, MA, Jun. 2010.

[77] M. A. Guvensan and A. G. Yavuz, "On coverage issues in directional sensor networks: A survey," *Ad Hoc Networks*, vol. 9, pp. 1238–1255, 2011.

[78] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout algorithms for combinatorial optimization," *Journal of Heuristics*, vol. 3, pp. 245–262, 1997.

[79] C. J. Costello and I. J. Wang, "Surveillance camera coordination through distributed scheduling," in *Proc. 44th IEEE Conf. Decision and Control*, Seville, Spain, Dec 2005, pp. 1485–1490.

[80] G. P. Kefalas, "A phased-array ground terminal for satellite communications," *IEEE Trans. Commun. Technol.*, vol. 13, no. 4, pp. 512–525, 1965.

[81] A. O. Hero, C. M. Kreucher, and D. Blatt, "Information theoretic approaches to sensor management," in *Foundations and Applications of Sensor Management*, A. O. Hero, D. Castanon, D. Cochran, and K. Kastella, Eds. New York: Springer, 2008.

[82] Z. Zhang, E. K. P. Chong, A. Pezeshki, W. Moran, and S. D. Howard, "Submodularity and optimality of fusion rules in balanced binary relay trees," in *Proc. 51st IEEE Conference on Decision and Control*, Maui, HI, Dec. 2012, pp. 3802–3807.

[83] Z. Zhang, Z. Wang, E. K. P. Chong, A. Pezeshki, and W. Moran, "Near optimality of greedy strategies for string submodular functions with forward and backward curvature constraints," in *Proc. 52nd IEEE Conference on Decision and Control*, Florence, Italy, Dec. 2013, pp. 5156–5161.

[84] S. Ragi, H. D. Mittelmann, and E. K. P. Chong, "Directional sensor control for maximizing information gain," in *Proc. SPIE 8857, Signal and Data Processing of Small Targets 2013*, San Diego, CA, 2013.

[85] ——, "Directional sensor control: heuristic approaches," submitted for publication.