2016

An event-driven approach to control and optimization of multi-agent systems

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

AN EVENT-DRIVEN APPROACH TO CONTROL AND OPTIMIZATION OF MULTI-AGENT SYSTEMS

by

YASAMAN KHAZAENI

B.S., Sharif University of Technology, 2005
M.S., West Virginia University, 2009

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2016

Approved by

First Reader

Christos G. Cassandras, Ph.D.
Professor of Electrical and Computer Engineering
Professor and Head of Systems Engineering

Second Reader

Ioannis Ch. Paschalidis, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Biomedical Engineering

Third Reader

Pirooz Vakili, Ph.D.
Associate Professor of Mechanical Engineering
Associate Professor of Systems Engineering

Fourth Reader

Sean B. Andersson, Ph.D.
Associate Professor of Mechanical Engineering
Associate Professor of Systems Engineering

To my family

Baba, Maman and Ali

بی شما من که هیچ

آسمان هم زمین می خورد

Acknowledgments

At first, I would like to express my most humble gratitude to my advisor, Prof. Christos G. Cassandras, without whom I would have never been able to be where I am right now. His sincere interest in my success, his patience, motivation, and incredible amount of knowledge have been the light on my way. His guidance helped me find my path in research in many ways. I could not have imagined having a better advisor and mentor for my PhD study.

Besides my advisor, I would like to sincerely thank the rest of my dissertation committee, Prof. Paschalidis, Prof. Vakili and Prof. Andersson. Thanks for your cooperation, guidance and most of all your ideas and questions. I also thank Prof. Caramanis for accepting to serve as the chair for my PhD defense.

During my PhD journey, everybody in the Division of Systems Engineering have been kind, respectful, and always available for help. I would like to specially thank Prof. Wang, Ms. Elizabeth Flagg, Ms. Cheryl Stewart, and Ms. Ruth Mayson for making everything possible in the past five years. I also thank the Center for Information and Systems Engineering for their great help throughout these years, specially I thank Ms. Denise Joseph and Ms. Christina Polyzos for all their help.

Having spent most of my time here at CODES lab during the past four and half years, I would like to thank my current and past labmates for all the moments we have spent together. For all the nights we spent on deadlines, the fun conversations, and for making this journey much more meaningful. Special thanks to Sepideh, Julia, Xuchao, Xinmiao, Yue, Nan and Rebecca. Also to the rest of my BU family, Setareh, Mohammad, Armin, Shahrzad, and Soudeh who were always up for a tea time and awesome conversations.

I also like to thank Elli and Evgeny for making my first year at BU the most memorable year ever. Cheers to all the nights we studied for the qualifying exams

which made us friends for life.

In all different stages of your life, you need those friends who would help you forget about all the stress and re-energize, I have been blessed with many and with all my heart I am grateful to them. Vida, Sheyda, Maryam, Hossein, Mohammad, Sadaf, Omeed, Shirin, Julia, Eugene and Amir, I couldn't be me without you all and if it wasn't for your support I would have never survived another five years of graduate school.

On a lighter note, I thank my two beautiful cats, Shooka and Keeja, who made sure to sit by my side all day and night when I was finishing up this dissertation.

Last but of course not least, I want to thank my dearest family. I am thankful to my Dad. *Baba*, I appreciate every single moment I have spent with you and I regret all those that I have missed. Your love keeps my heart beating. I am for ever thankful to my Mom. *Maman*, You have been what I have always wanted to be. Your love and friendship are my most valuable assets. Thanks for asking me to calculate the sum of one to ten when I was nine and thanks for teaching me math for it is the purest joy of my life.

I am and will always be indebted and thankful to Ali for his support, presence and love. You have the most lovable soul and the most loving heart. We made it here and it was only possible together. I love you.

AN EVENT-DRIVEN APPROACH TO CONTROL AND OPTIMIZATION OF MULTI-AGENT SYSTEMS

YASAMAN KHAZAENI

Boston University, College of Engineering, 2016

Major Professor: Christos G. Cassandras, PhD
Professor of Electrical and Computer Engineering,
Professor and Head of Systems Engineering

ABSTRACT

This dissertation studies the application of several event-driven control schemes in multi-agent systems. First, a new cooperative receding horizon (CRH) controller is designed and applied to a class of maximum reward collection problems. Target rewards are time-variant with finite deadlines and the environment contains uncertainties. The new methodology adapts an event-driven approach by optimizing the control for a *planning horizon* and updating it for a shorter *action horizon*. The proposed CRH controller addresses several issues including potential instabilities and oscillations. It also improves the estimated reward-to-go which enhances the overall performance of the controller. The other major contribution is that the originally infinite-dimensional feasible control set is reduced to a finite set at each time step which improves the computational cost of the controller.

Second, a new event-driven methodology is studied for trajectory planning in multi-agent systems. A rigorous optimal control solution is employed using numerical solutions which turn out to be computationally infeasible in real time applications. The problem is then parameterized using several families of parametric trajectories.

The solution to the parametric optimization relies on an unbiased estimate of the objective function's gradient obtained by the "Infinitesimal Perturbation Analysis" method. The premise of event-driven methods is that the events involved are observable so as to "excite" the underlying event-driven controller. However, it is not always obvious that these events actually take place under every feasible control in which case the controller may be useless. This issue of event excitation, which arises specially in multi-agent systems with a finite number of targets, is studied and addressed by introducing a novel performance measure which generates a potential field over the mission space. The effect of the new performance metric is demonstrated through simulation and analytical results.

Contents

List of Tables

List of Figures

List of Abbreviations

CRH	.....	Cooperative Receding Horizon
DES	.....	Discrete Event Systems
DP	.....	Dynamic Programming
DVRP	.....	Dynamic Vehicle Routing Problem
HS	.....	Hybrid Systems
IPA	.....	Infinitesimal Perturbation Analysis
KP	.....	Knapsack Problem
MPC	.....	Model Predictive Control
MRCP	.....	Maximum Reward Collection Problem
OP	.....	Orienteering Problem
PMP	.....	Pontryagin's Maximum Principle
RHC	.....	Receding Horizon Control
TPBVP	.....	Two Point Boundary Value Problem
TSP	.....	Traveling Salesman Problem
UAV	.....	Unmanned Aerial Vehicle
VRP	.....	Vehicle Routing Problem
WSN	.....	Wireless Sensor Network

Chapter 1

Introduction

In this chapter, we will go through some fundamental problem of cooperative control and then move to basics of time-driven and event-driven control methods. Afterward we introduce some background on methods that have been used in this dissertation such as Receding Horizon Control and Optimal Control Methods. Later, we introduce the two main problems discussed in the dissertation. This chapter is ended by providing the main contributions of this work.

1.1 Cooperative Control

Cooperative Control deals with systems that are characterized by a set of interconnected decision-making components with limited storage and processing capabilities, that can provide locally sensed information and limited inter-component communications, all seeking to achieve objectives defined globally or individually (Shamma, 2007), (Murray, 2007), (Murphey and Pardalos, 2002).

Cooperative control problems appeared in the research areas of military systems and got expanded into mobile sensor networks, manufacturing, transportation systems, smart cities and other network problems. Flight vehicle formation (Fax and Murray, 2004), (Fowler and D'Andrea, 2002), UAV cooperative control (Parker, 1993), swarm formation (Olfati-Saber, 2007), (Bayindir, 2016),(Tanner et al., 2007), cooperative classification and surveillance (Chandler et al., 2001), mobile agents coordination (Jadbabaie et al., 2003), rendez-vous problems (McLain et al., 2001), (Yao

et al., 2010), persistent monitoring (Cassandras et al., 2013) and coverage control (Zhong and Cassandras, 2011), (Schwager et al., 2009), (Cortes et al., 2004), are some well known examples. Another problem that has been extensively looked at is the consensus problems where the network entities are supposed to reach an agreement based on a distributed set of information which normally is only partially available to each of them at the beginning of the problem (Tahbaz-Salehi and Jadbabaie, 2008), (Ren and Beard, 2008), (Ren et al., 2005a).

In any cooperative control problem, the controllable members of the network are responsible for the cooperation. This cooperation might be through carrying information and resources or performing tasks within the network. These entities can be autonomous vehicles or UAVs, mobile robots acting, mobile sensors as message ferries, etc. In a general framework we call these cooperating members, "agents" and the problems that involve controlling a network of these agents are called *multi-agent cooperative control problems*. In most cases, problems have points of interests within the problem environment which can be locations that need to be visited, sensors or data sources that should be collected or a formation trajectory that should be followed by vehicles. We call these points of interest, "targets". Targets may also carry information or they may be moving, but they don't actuate any control unlike the agents that are responsible for actuation of the control. The solution to these problems is to find the control for all the agents that optimizes the global/individual objectives involving the targets of the problem.

Optimizing a multi-agent system, is to find the best agents' state to optimize the system's performance measure that describes the interaction of these agents with the environment. These optimizations are either static when a single state value (i.e. location) of the agents need to be found. An example for such problem are task allocation where multiple agents are going to be assigned to a number of tasks by

optimizing a certain total reward (Panagou et al., 2014), or coverage control problem where optimal location of agents are calculated to optimally cover a bounded environment (Sun et al., 2014). On the other hand, the dynamic multi agent problems are when, the state of the agents is optimized for a period of time which can be finite or infinite. These can come down to many types of trajectory optimization problems such as persistent monitoring, data harvesting, robot's trajectory planning around obstacles, and many other such problems that were mentioned before. The static problems also can have a further dynamic optimization step that designs the path for the mobile agents to pursue the task allocation, etc.

The general setting of multi-agent systems gives rise to a complex stochastic system that can be solved in a centralized or decentralized fashion. In the centralized methods it is only in the execution level that agents collaborate and the decision making happens by a central computer while all the system information is known to the central station and no communication is needed between the agents. Every agent's controls are calculated by the central computer. The agents would only actuate the control they receive from the central station.

In the centralized approach a global optimization problem is formulated where the total objective function takes into account every agent's contribution. This global optimization problem is then solved for optimal control values for all agents. This optimization problem is normally is a complex nonlinear problem which is computationally expensive to solve. Solution methods can be based on non-linear programming (Raghunathan et al., 2003), game theoretic frameworks (Harmati and Skrzypczyk, 2009), or semi-definite programming (Frazzoli et al., 2001).

The computation cost of the centralized algorithm can grow exponentially with the number of agents and they become more expensive in terms of computation and memory (Defoort et al., 2009). Despite these drawbacks, for systems where

the agents don't have computing capabilities, or systems where synchronicity is very important we are better off with centralized methods and have to find solutions that can overcome the computational burdens.

On the other hand, in the decentralized methods, agents can collaborate in decision makings through a distributed system and each agent has a memory and computation capacity. The agent is capable of calculating its own control given that it receives enough information from its neighbors. Communication between the agents is the key difference between the centralized and decentralized cooperative control methods. Identification of the pieces of information to be shared between agents is a critical step in the formulation of a decentralized approach. In these algorithms, the state of the problem is only partially known by each agent (Ren et al., 2005b). (Ren, 2006), (Li and Cassandras, 2006a), (Cruz et al., 2007).

In (Ren, 2006) the centralized and decentralized implementations of a consensus based algorithm are compared for problems such as cooperative timing, formation maintenance, rendez-vous, altitude alignment, and synchronized rotations where it's shown decentralized schemes are superior to centralized schemes in terms of robustness and scalability.

1.2 Time-driven and Event-driven Control

The fundamental view of dynamical systems has mainly been based on a time-driven approach which is rooted in the theory of differential equations. We basically hypothesize a "clock" for the system where every "clock-tick" updates the state of the system. The theories that come after for sampling, estimation, control and optimization of the dynamical systems are based on this event-driven approach. The clock-tick synchronizes all the components of the system and updates are recorded even when no changes occur at each tick.

While this has historically resulted in many advancement in control and optimization designs, by moving to wireless, networked, and distributed complex systems; the applicability of the time-driven approach falls short in some areas. While it is close to impossible to hold the synchronicity for all the components, it also is not efficient to trigger actions with every clock-tick when such actions may be unnecessary. On the other end, event-driven approaches offer an alternative view to the modeling, control, communication, and optimization of dynamical systems.

The general idea behind the event-driven methods is that the system at hand doesn't necessarily need actions to be taken at each clock-tick. By accurately understanding the systems dynamics, one can identify proper events that cause changes in the dynamics and call for actions to be taken. One can also note that this new paradigm can include the traditional time-driven view if the clock-tick is considered a proper event in the system. Defining the right events is a crucial modeling step and has to be done with great understanding of the system.

The foundation of event-driven methods is mostly understood in studying the theory of discrete event systems(DES) and Hybrid Systems(HS) (Cassandras and Lafortune, 2006), but more recently there have been significant advances in applying event-driven methods (also referred to as "event-based"and "event-triggered") to classical feedback control systems; e.g., see (Heemels et al., 2008), (Anta and Tabuada, 2010), (Trimpe and D'Andrea, 2014). In the distributed networked system the idea of event-driven control allows for less communication while achieving same performance goals if designed properly, see (Miskowicz, 2015) and (Cassandras, 2014) and references therein. Event-driven approaches are also attractive in receding horizon control, where it is computationally inefficient to reevaluate a control value over small time increments as opposed to event occurrences defining appropriate planning horizons for the controller e.g., see (Li and Cassandras, 2006b), (Khazaeni and Cassandras,

2014).

One main area of this application is in the decentralized or distributed control systems where the event driven method allows for an event-driven communication between the agents of the system instead of synchronous communication protocols (Zhong and Cassandras, 2010). This asynchronous, event-driven communication can help saving the energy of the agents specially when they are sensors with low power capacity in a large sensor network or it can create a more efficient communication when there is limited wireless communication capability in the system. A comparison of time-driven and event-driven control for stochastic systems in favour of the latter is found in (Astrom and Bernhardsson, 2002).

Uncertain environment and dynamics in multi-agent systems adds anther layer of complexity when one tries to optimize these systems. In these problems, the dynamic of the agents and targets can be uncertain. In (Yucelen and Johnson, 2012) the agents are autonomous vehicles with uncertain dynamics where a new vehicle-level decentralized robust adaptive control approach is introduced to suppress the effect of nonlinear uncertain dynamics of the vehicles. In (Zeng-Guang et al., 2009) a decentralized robust adaptive control approach is introduced for the consensus problem of multi-agent system. The uncertainties of the vehicle dynamics are addressed by using adaptive neural network and robust control techniques. Other uncertainties can be a result of the problem environment like obstacles that are unknown or threats that can appear during the course of the problem and should be avoided by the agents; see (Blackmore et al., 2011), (Deittert et al., 2010), (Polycarpou et al., 2001), (Burgard et al., 2002). Targets that have uncertain dynamic and behavior like random times of arrival can be another source of uncertainty, (Li and Cassandras, 2006b).

Modeling the uncertainties can be different depending on the type of information at hand. Many works use probabilistic models and use methods such as Bayesian

filtering at each iteration to improve the prior information and basically learn the system. In (Furukawa et al., 2006) a coordinated control technique is built using heterogeneous vehicles to autonomously search for and track multiple targets using recursive Bayesian filtering. They use a grid-based probability density function (PDF) to represent target location in the space.

In the uncertain environment, where new information about the environment might become available at any time and no probabilistic information is available *a prioi*, we can see each new piece of information as a random event in the problem environment like changes in the location or characteristic of a target, or a failure in one of the agents happening at a random time. This allows one to pro-actively respond to random events without any prior information that is being learned throughout the course of the mission. Such examples are shown to work effectively in some cases where no information is available on the arrival time of targets, see (Li and Cassandras, 2006b), (Khazaeni and Cassandras, 2014).

In this approach, when an event happens, a new set of controls need to be calculated for all the agents in contrast to the time-driven view where the new controls are calculated periodically with a synchronous clock. One can also define, artificial timeout events to ensure the control is re-evaluated frequent enough in case no new information is becoming available. This can help in ensuring the stability of the controller. In (Demir and Lunze, 2012) authors propose an event-driven design for a multi-agent systems controller. It is shown that the system behavior with continuous state-feedback controller can be approximated with this event-driven controller for any arbitrary precision. The performance of the event-driven controller is evaluated by comparing the event-driven control loop with the continuous state-feedback loop.

1.3 Receding Horizon Control - an Event-driven Approach

In multi-agent systems optimization the resulting problem is high dimensional complex optimal control problem where the solution can be computationally intractable. The common technique that is used in solving problems with this level of complexity in *real time* is to divide the problem into sub-problems. The first view is a functional decomposition where the complete complex problem is decomposed into sub-problems with less complex objectives and possibly fewer variables (Bellingham et al., 2002), (Finke et al., 2003). In (Earl and D'Andrea, 2007) a decomposition method is introduced to solve the cooperative control problems in multi-vehicle systems. The problem is divided into a task assignment and a task completion sub-problems. The task completion is a dynamic control problem where the optimal control to complete the assigned task for each vehicle is calculated considering its constraints. The task assignment problem which is a combinatorial problem is solved using a branch and bound algorithm. The alternative to this functional decomposition is a time decomposition approach. The main idea is to solve an optimization problem seeking to maximize the total expected reward accumulated by the network over a given time horizon, and then continuously extend this time horizon forward (either periodically or in purely event-driven fashion). This idea, introduced in (Cassandras and Li, 2002), is in the spirit of receding horizon (RH) schemes, which are associated with model-predictive control and used to solve optimal control problems for which feedback solutions are extremely hard or impossible to obtain (Cassandras and Li, 2005).

Model predictive control (MPC) or receding horizon control (RHC) is a control solution method in which the current control action is calculated by solving a finite horizon open-loop optimal control problem, at each sampling instant, using the current state of the system as the initial state. This is an online time-driven solution

where new control values are calculated at time steps defined by the clock of the system and the length of the time horizon of the specified optimal control problem. The optimization yields an optimal control sequence from which the first control is applied to the system. This is followed by a time step for which the control value is maintained and then a sampling instant with a new optimal control problem and new initial state. This is its main difference from conventional control which uses a pre-computed control law. RHC enables us to solve problems where calculation of a control law is difficult or impossible due to the complexity and uncertainty of the problems and every state of the system has a different set of constraint on the control (Mayne et al., 2000).

A standard optimal control problem is solved at each step of the receding horizon control, except that it has a finite horizon in contrast to the infinite horizon problems in H_2 and H_∞ linear optimal control. The RHC can provide on-line solution of the optimal control problem for the current state of the system, rather than determining a feedback policy that provides the optimal control for all states. The on-line solution is obtained by solving an open-loop optimal control problem where the initial state is the current state of the system. Determining the feedback solution, on the other hand, requires solution of the Hamilton-Jacobi-Bellman (Dynamic Programming) differential or difference equation which is more difficult.

In (Mayne and Michalska, 1990) the receding horizon control method is shown to have stabilizing results for non-linear control system problems. Also in (Dunbar and Murray, 2006) a distributed approach is formulated for receding horizon control of multi-vehicle problems. The control problem is decoupled into sub-systems with independent dynamics and constraints while the state of the subsystems is considered to be completely coupled. The control updates for all vehicles is synchronous and the receding horizon step should be small enough to insure the stability of the controller.

The subsystems are assumed to have complete knowledge of the prior state trajectory of the other subsystems.

In most of the RHC classic applications the implementation is a time based approach where the finite time horizon is a fixed value. In (Chen et al., 2010) a time-driven receding horizon is presented for a multiple mobile formation control using a leader-follower scheme. In (Izadi et al., 2012), a new RHC algorithm for a group of cooperative vehicles is investigated where the communication bandwidth is limited. The result is a decentralized RHC with communication delays. A new approach is proposed to find the communication bandwidth for each vehicle, subject to network bandwidth constraints, in order to improve the cooperation performance. The proposed bandwidth allocation approach is decentralized and does not require significant online computation and communication resources.

The idea of an event-driven controller for the uncertain multi-agent system is to solve a new optimal control problem with a dynamic finite horizon when a new event happens in the system. The events should be precisely defined based on the topology of the problem and its uncertainties. Changes in the agents' dynamic, appearing/disappearing of points of interest in the environment, recognizing a threat, etc are examples of these "events". The non-periodic finite horizon at each step of the problem needs to be defined based on the dynamics and topology of the problem. This horizon is dynamically updated if a random event is detected while the controls are executed.

A schematic explanation of this is shown in Fig. 1·1. In this figure the lower time axis shows the time-driven receding horizon control. At each sampling time t_i a new optimal control problem with a time horizon of H_i is solved to find a control sequence for that time. The first value of that sequence is called u_i and is maintained for the interval of t_i to t_{i+1}. At this time the process repeats and a new control

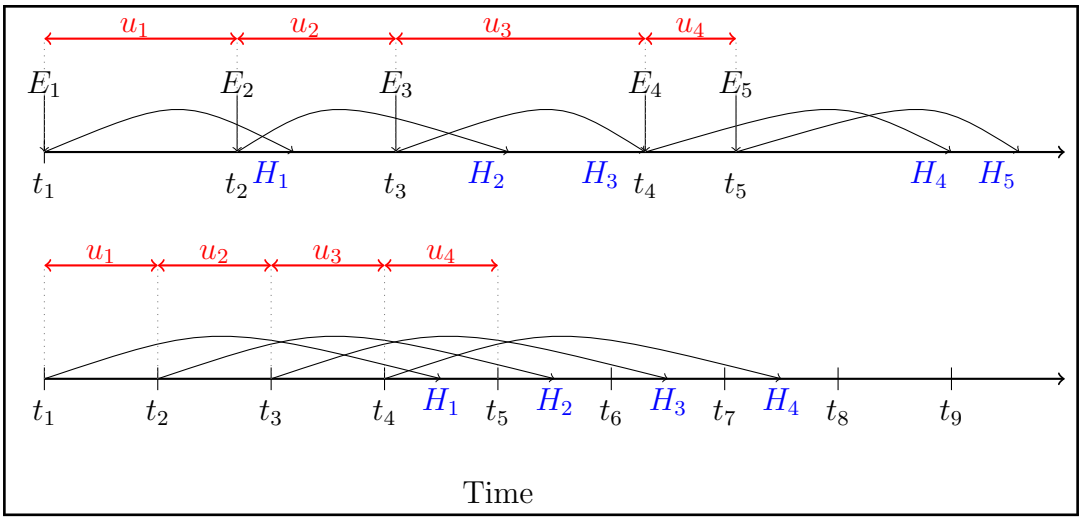


Figure 1·1: Time-driven and event-driven receding horizon

is calculated. In the upper time access, an event-driven receding horizon control scheme is illustrated. A new event E_i happens at time t_i and a new finite horizon H_i is calculated. A new optimal control with finite horizon H_i is then solved to find the control value u_i which is maintained for the duration of the horizon or until a new event happens, whichever comes first.

As an example, an event triggered cooperative control is used for the consensus problem in (Dimarogonas and Johansson, 2009) with centralized and decentralized approaches. In (Cassandras and Li, 2002), (Li and Cassandras, 2006b), (Yao et al., 2010) and (Tomasson, 2011) centralized and decentralized receding horizon control algorithms are applied to different cooperative control problems such as maximum reward collection, data harvesting and resource allocation. In most of these applications the receding horizon control provides a trade off between the long-term against short-term decisions in the presence of uncertainties. The finite time horizon at each step is not of the same length and different stochastic events trigger a new time step.

1.3.1 Maximum Reward Collection Problem

The first problem that is considered in this dissertation is the maximum reward collection problem (MRCP) with time varying rewards. The goal of the problem is for N agents to collect time dependent associated "rewards" from "M" stationary targets in a 2-D environment. The problem can involve uncertainties rooted in the dynamics of the agents, appearance and disappearance of targets and their location or obstacles and threats in the problem environment. In the problems considered in this work targets carry time varying non increasing rewards and can appear or disappear at any time. An MRCP *mission* is defined as collecting the maximum possible reward from M number of targets by N number of agents in the minimum amount of time or by a pre-defined deadline T.

In the deterministic environment if targets carry constant equal rewards, a one agent MRCP is an instance of Traveling Salesman Problem , (Salz, 1965) and (Salz, 1966). A Traveling Salesman Problem (TSP) is where one salesman has to visit multiple cities once and only once, starting and ending at a depot. One can also define an m-TSP in which m salesman are cooperating to visit the cities, meaning each city should be visited once and only once by exactly one salesman. This is also called the Vehicle Routing Problem (VRP) which is a very important problem in the field of distribution, transportation and logistics, (Laporte, 1992),(Dantzig and Ramser, 1959). As a fundamental definition a VRP is the problem of "finding a set of routes for K identical vehicles based at the depot, such that each of the vertices is visited exactly once, while minimizing the overall routing cost" (Pillac et al., 2013). In general form of this problem a number of vehicles with certain capacities should gather rewards or deliver resources to customers with different values or demands. Vehicles might have a specific time window that they can be used during that. This is called the Vehicle Routing Problem with Time Windows. Demands or targets

might also be dynamic which results to problems that are generally called "Dial A Ride Problem" and modeled as Dynamic Vehicle Routings (DVRP) .

TSP and VRP are both combinatorial problems for which finding the global optimum is the solution of a complex combinatorial optimization. There is a vast number of methods to solve the TSP by exact and approximate algorithms (Applegate et al., 2011), (Arora, 1998). The exact solutions are normally found through complete permutations or integer programming formulation of the problem which is solved by decomposition and branch and bound methods. These methods are computationally intensive and inhibitive for very large problem. The most well known heuristic algorithms for TSP are the Christofides' Algorithm (Christofides, 1976) and Lin-Kernighan (Lin and Kernighan, 1973). Very close bounds have been found by highly efficient heuristic solvers for big instances of the problem. Instances of more complicated TSP formulations with constraints on the capacity of the vehicles and unequal demands in the targets have been studied in the literature (Hernandez-Perez and Salazar-Gonzalez, 2014) however, most of these methods would solve the problem when no uncertainty is involved.

The VRP and DVRP literature is extensive. In (Pillac et al., 2013), a comprehensive review is provided which points to the computational complexity of these problems. Beyond dynamic programming, various heuristics are described, including methods such as Genetic Algorithms and Ant Colony Systems for different versions of DVRPs. A broad taxonomy of solution approaches may be found in (Lahyani et al., 2015). In (Bullo et al., 2011), a variety of VRPs is considered from a queueing theory point of view and solution algorithms are given that provide some performance guarantees. In (Ekici and Retharekar, 2013), (Tang et al., 2007), a deterministic MRCP with a linearly decreasing reward model is cast as a dynamic scheduling problem and solved via heuristics.

The uncertainties in the location of targets (cities) or dynamically arriving customers with varying demands can cause great inefficiencies in the TSP and VRP solvers. Appearance of a new target can completely change the course of the solution and a new instance of the problem needs to be solved.

The other class of problems is the Orienteering Problems (OP) rooted in the sport game of orienteering (Chao et al., 1996b). In this game, each player starts at a specified control point, visits as many checkpoints as possible and returns to the control point within a given time frame. Each checkpoint has a certain reward and the game's objective is to maximize the total collected reward. The setup of the game is as such that the player needs to decide to visit the most rewarding checkpoints in a bounded time. (Vansteenwegen et al., 2011) provides a survey on the different algorithms in the literature for the OP. The individual problem is a combination of knapsack problem (KP) (Papadimitriou and Steiglitz, 1998) and the traveling salesman problem (TSP). A mixed integer linear programming problem is formulated for the OP where branch and bound and branch and cut are proposed for this problem to provide upper bounds. A number of heuristic algorithms are also discussed. In (Chao et al., 1996a) authors discuss a team orienteering problem where they propose a heuristic solution for that. The problem is based on the orienteering game but when multiple players are performing as a team. The heuristic solution first assigns a set of control points to each team member and then by adding and exchanging points between members maximizes the total reward within the time limit. In both OP and Team OP the assumptions are that the reward at each control point is not changing and the position of those points are known a priori.

In (Bansal et al., 2004), authors model the OP and Team OP as a Deadline-TSP problem on a graph where the objective again is finding a path starting at one node and visits as many nodes as possible by their deadlines. They propose an

$O(\log n)$-approximation algorithm for the Deadline-TSP problem based on an approximation of the point-to-point orienteering problem. They also provide an $O(\log^2 n)$-approximation for the Vehicle Routing Problems with Time Windows, in which the vehicles availability is a time window. Underlying assumptions like many path planning problems are that the targets are known and agents can only move on the edges of the graph.

Because of the MRCP complexity, it is natural to resort to decomposition techniques. One approach is to seek a functional decomposition that divides the problem into smaller sub-problems which may be defined at different levels of the system dynamics, see (Bellingham et al., 2002), (Earl and D'Andrea, 2007). An alternative is a time decomposition where one can use receding horizon techniques. In the context of multi-agent systems, a Cooperative Receding Horizon (CRH) controller was introduced in (Li and Cassandras, 2006b) with the controller steps defined in event-driven fashion (with events dependent on the observed system state) as opposed to being invoked periodically, in time-driven fashion. The method is extended into a graph representation with a switching CRH controller in (Chini et al., 2014). A decentralized version of the CRH controller is also introduced in (Li and Cassandras, 2006a). A key feature of this controller is that it does not attempt to make any explicit agent-to-target assignments, but only to determine headings that at the end of the current planning horizon, place agents at positions such that a total expected reward is maximized. Nonetheless, as shown in (Li and Cassandras, 2006b), a stationary trajectory for each agent is guaranteed under certain conditions, in the sense that an agent trajectory always converges to some target in finite time.

The maximum reward collection problem with linearly decreasing rewards in deterministic environments has been previously tackled in (Ekici and Retharekar, 2013), (Tang et al., 2007) where heuristic methods are used to solve the problem as

a dynamic scheduling. The problem can be viewed as a discounted reward Traveling Salesman Problem (TSP) and is the modeling platform for many of the cooperative control problem that were discussed before.

Even though we refer to the targets as reward bearing points, the same approach can be used in defining problems in applications such as disaster relief where the nodes can be thought of as hazardous areas from which people have to be evacuated. The problem then becomes one of assigning vehicles to areas to collect people and transport them back to safety at the bases.

1.4 Event-driven Control of Multi-agent Systems

1.4.1 Data Harvesting Problem

Advances in wireless communication, embedded powerful controllers and small and inexpensive sensors over the recent years have made wireless sensor networks(WSN) an applicable tool in many applications. A wireless sensor network is a collection of sensors, able to communicate between each other to perform exploration, monitoring and surveillance. WSNs and conventional communication network are different in design, types of equipment, and their performance metrics. The sensors in a WSN might be mobile and able to execute functions during the sensing process. The limited computation and power capabilities in the wireless sensor networks makes the lifetime of the nodes be a main concern when designing control methods for them. The main approach has been to deploy a few powerful rechargeable mobile sensors in a network. The availability of small, cheap and non-rechargeable sensors have changed this approach to much bigger networks with less capable nodes. A tutorial-style overview of sensor networks from a systems and control theory perspective is presented in (Cassandras and Li, 2005). Developments in the mobile sensors, has been followed with abundance of cheaper drones and UAVs that are available to public for all

sorts of applications. They can also form networks and perform sensing, information collection and even delivery of goods.

Although these electronic devices are capable of many tasks and inexpensive, they have limited capabilities in communication and power. These limitations encourage reducing the energy consumption of individual device to maximize their lifetime. We can generally refer to these as "mobile agents" since the mobility and controllability is a common feature among all of them in different application.

Once the task at hand is to be accomplished by a group of mobile agents, the cooperation level is determined by the type of global and local performance measures defined for the whole system and each agent. These measures might be only on a global level specially when the control is going to be calculated in a centralized fashion, can be at the agent's level in the fully distributed control or a combination of both.

Data harvesting and its variation minimum latency problem (Blum et al., 1994) is the problem of gathering the data from a stationary point of interests, called "targets", where a direct communication path does not exist between each data generating node and the central sink or base node. Although defined as stationary points, targets can be assumed non-stationary in the data harvesting problem as long as some knowledge of their movements is available. This obviously created more complexity in the control and optimization of the system. In a sensor network, the functional life span increases when there is more delay in delivering the data from the network nodes (Tekdas et al., 2009), (Wei et al., 2008). This delay tolerant system needs a control scheme to deploy the mobile nodes to gather the available data with the least delay. These mobile nodes are called message ferries or simply ferries. The mobile nodes visit the data generation nodes and collect the data and deliver it to the base. They might have limited buffer sizes that needs visits to the base once the buffer is full.

Systems consisting of cooperating mobile agents have been continuously developed for a broad spectrum of applications such as environmental sampling (Corke et al., 2010),(Smith et al., 2011), surveillance (Tang and Özgüner, 2005), coverage (Zhong and Cassandras, 2011),(Chakrabarty et al., 2002),(Cardei et al., 2005), persistent monitoring (Alamdari et al., 2014),(Cassandras et al., 2013), task assignment (Panagou et al., 2014), and data harvesting and information collection (Klesh et al., 2008),(Ny et al., 2008),(Moazzez-Estanjini and Paschalidis, 2012).

Examples of the data harvesting application include environmental sensors (Hart and Martinez, 2006), sensors for monitoring air/water quality, traffic meters, machinery condition monitoring, utility meters, etc. In many, sensors are sparsely distributed, energy limited or bandwidth capacitated so they cannot afford long-range wireless communications. More applications can be found in (Zhao et al., 2005) and (Pandya et al., 2008).

There are also equivalents to this problem outside the sensor network realm, such as in disaster planning, evacuation process and rescue operations, pickup/delivery and transportation systems, surveillance operations using drones and UAVs. The general theme is a network of mobile agents need to visit points of interests during the course of the problem and perform visits to the base on a frequent basis. The base visits can be for delivery of data/goods/people or recharge or renew power supply/fuel. For example, the flying time span of a drone on one battery charge is limited so the flight trajectories need to be optimized and returns to base might be scheduled for recharge or loading/unloading.

In the data harvesting problem mobile agents are sometimes known as "message ferries" or "data mules"Ḣaving its root in the wireless sensor networks, the problem is normally studied on a directed or undirected graph where minimum length tours or sub-tours are to be found. The graph topology view of the problem utilizes many

routing and scheduling algorithm from wireless sensor networks, see (Akkaya and Younis, 2005),(Liu et al., 2011) and references therein. The main advantage of the graph topology is the ability to adjust to environment constraints. Movements inside a building or within a road network are examples for which a graph topology is very suitable. These methods have several drawbacks, they are generally combinatorially complex, they treat agents as particles (hence, not accounting for limitations in motion dynamics which should not, for instance, allow an agent to form a trajectory consisting of straight lines), and they become computationally infeasible as on-line methods in the presence of stochastic effects such as random target rewards or failing agents since the graph topology has to be re-evaluated as new information become available. As an example, in (Chang et al., 2014) algorithms are proposed for patrolling target points with the goal of balanced time intervals between consecutive visits. A weighted version of the algorithm improves the performance in cases with unequally valued targets. However, in this scenario the data need not be delivered to a base and visits to a recharging station are only necessary if the data mules are running out of energy.

On the other hand, the problem can be viewed as 2-D or 3-D trajectory optimization problem, where the mobile agents are freely (or with some constraints) moving in the space and visits the targets by getting close to them with some criterion. For example the targets can be assumed to have a sensing range within which the mobile agent can initiate a wireless communication with them and exchange data. In problems where a physical visit is needed such as rescue missions, the mobile agent has to visit the exact target points and perform the task. These trajectories not necessarily consist of straight lines as opposed to in the graph. The trajectories can adjust to uncertainty in the target locations when the exact locations are not necessarily known. Also, adjusting to limitation in agent's mobility is another advantage. Constraining

trajectories to obstacles or environment's boundaries might not be as straight forward but is possible. In (Ny et al., 2008) the problem is viewed as a polling system with a mobile server visiting data queues at fixed targets. Trajectories are designed for the mobile server in order to stabilize the system, keeping queue contents (modeled as fluid queues) uniformly bounded.

Another benefit of modeling the problem as a trajectory optimization is the ability to parameterize the trajectories with different type of functional representations. The parametric class that is considered for each problem might be different but results in optimization problems with fewer number of controls. If the parametric trajectory family is broad enough, we can recover the true optimal trajectories; otherwise, we can approximate them within some acceptable accuracy. Moreover, adopting a parametric family of trajectories and seeking an optimal one within it has additional benefits: it allows trajectories to be periodic, often a desirable property, and it allows one to restrict solutions to trajectories with desired features that the true optimal may not have, e.g., smoothness properties to achieve physically feasible agent motion. In (Lin and Cassandras, 2015), the parametric trajectory planning method is used for a persistent monitoring problem. In this work, the problem is cast as an optimal control resulting in a two point boundary value problem. Solving the TPBVP provides some structures for the optimal control policy which is then used in the parametric trajectory design which can solve larger instances of the problem compared to the TPBVP.

1.4.2 Optimal Control Methods

Optimal Control theory is the optimization method that deals with finding control policies for a control system for which the optimality criterion is defined through a cost functional. By minimizing the cost that is a function of the control variables and the state of the system one can achieve control policies that satisfy an optimality

criterion (Pontryagin, 1987).

In general when the optimization problem is defined for a continuous time domain the main challenge is that the size of the control policy is infinite. In addition at each time instance there might be an infinite size feasible control set that one can choose from. This calls for rigorous mathematical methods. On the other hand, one can discretion the problem into time intervals and find policies that provide a control value for each time interval. These cases might be solved with conventional nonlinear optimization methods.

As in most cases the optimal control the cost functional is highly nonlinear, analytical solutions do not exist for the problem. In these cases numerical methods should be used to obtain a solution. Pontryagin's Maximum Principle (PMP) (Bryson and Ho, 1975) provides a necessary condition to solve for the optimal control policy. Calculus of variation which is the basics of optimal control theory can be used to solve for the interior solutions but in many applications the decision variables are bounded. PMP deals with these cases and uses the Hamiltonian analysis by minimizing the Hamiltonian function over all feasible controls. This optimization problem results into a Two Point Boundary Value Problem (TPBVP) for which the state and costate are known at initial and final times respectively.

In the Data Harvesting problem which was introduced, we use optimal control method to initialize our analysis for a deterministic setting. A TPBVP is defined and solved numerically. The solution provides special structure on the control of the agents in the environment. Studying the deterministic problem leads to understandings of the control policy structure that can be further extended to a stochastic environment.

1.5 Thesis Contributions

1.5.1 Event-driven Receding Horizon Control of Multi-agent System

In this thesis, for the first part we focus on application of the event-driven Receding Horizon Control method in the control of multi-agent systems. We design and implement a new Cooperative Receding Horizon (CRH) controller for a class of cooperative multi-agent systems continuing on some of the ideas in (Li and Cassandras, 2006b). The proposed controller design trajectories for a set of agents to cooperatively perform a maximum reward collection from a set of stationary targets with random arrival times and time-variant rewards.

Multi-agent systems which can be considered within this framework are cooperative path-planning, resource allocation, reward maximization, data harvesting, etc. All of these commonly require solutions that are on-line specially in situations with limited computation capacity, uncertain environment and real-time constraints.

We propose a centralized cooperative receding horizon control method that is based on an event-driven time decomposition of the control problem. We apply this scheme to the Maximum Reward Collection Problem (MRCP) which was put forward before. The problem environment is assumed to be uncertain with possible new targets arriving at any time, the agents can have a limited sensing range and they may detect targets at certain distances. The modeling framework can also account for threats, obstacles and agent's failures during the mission.

The controller solves a finite horizon optimization problem that maximizes the total expected reward collected by all agents at the end of the mission, while moving forward the finite horizon and re-solving a new instant of the optimization problem. We consider point-like agents that have a simple dynamic and move with a constant speed at all times. The controller determines agent's trajectories by calculating the heading for each agent at any time.

In the spirit of receding horizon methods, the optimal control is calculated for a *planning horizon* and is executed for a shorter *action horizon*. This action horizon is dynamically changing to account for new events happening during the mission. This fact results in a new optimization problem to be solved at any time a new piece of information becomes available. This enables us to handle problems with uncertain environment where new information is due anytime during the mission with no prior information on the arrival time of the information.

The CRH controller's performance measure at each time step is the expected total collected reward by the end of the mission. The mission as mentioned before can end at a specific time T, or when no more target is available. It should be noted that the optimization problem involved does not attempt to make any explicit agent-to-target assignments, but only to determine headings that, at the end of the current planning horizon, would place the agents at positions such that a total expected reward is maximized.

The controller in this work is built on the basis of the previous results from (Li and Cassandras, 2006b). Our new methodology enhances the performance of the previously designed controller while addressing some systematic shortcomings of the previous one. The first shortcoming of the previous CRH controller is that the expected collected reward in the performance measure is calculated as an unattainable loose upper bound. Meaning at each time step the controller assumes agents would reach every target through the minimum distance and minimum time. This yields to a loose lower bound on the expected visiting time for each target and an upper bound on its reward. In the present work, we try to improve this by using an estimated collected reward through an estimated path for each agent. We assume each agent would visit the targets in a shortest path order in which the metric for distance is a novel *travel cost factor*. This factor will be defined based on each targets reward,

its decline rate and the concentration of reward around it. The modified expected reward estimation improves the final result of the maximization problem significantly.

The new CRH controller also handles instabilities that were observed in the previous work (Cassandras and Li, 2002). There is still no direct target to agent assignment in this method however, at each time step a set of targets are defined as *active targets* for each agent. These will be the potential next visits for that agent. The active target set would be re-calculated at any event in the system so we ensure no target assignment before the agent is very close to a target where at that point the active target set becomes a singular set.

As a main outcome of the new design, in the proposed CRH control scheme, the feasible set for the heading of all agents at any time is reduced to a discrete set of headings. This feature reduces the optimization problem to a simple comparison over a finite set of controls. We prove that given our specific expected collected reward in the objective function definition, the optimal decision for the agents is to move toward one of the "active targets".

It is possible to extend the same framework to similar cooperative control problems such as resource allocation and data harvesting problems. In the resource allocation problem which is an instance of VRP/DVRP which were introduced previously, points of interests in the mission space have demands that should be satisfied by the resources located in a depot. Vehicles with limited capacity would take the resources to the demand points. Uncertainties such as random arrival of demands, unknown demands amount, agent failure, obstacles and other random event might be handled similar to the maximum reward collection problem.

1.5.2 Event Excitation in Multi-agent Systems

The premise of event-driven methods application in any multi-agent optimization problem is that the events involved are observable so as to "excite" the underlying

event-driven controller. However, it is not always obvious that these events actually take place under every feasible control: it is possible that under some control no such events are excited, in which case the controller may be useless. In such cases, one can resort to artificial "timeout events" so as to eventually take actions, but this is obviously inefficient. Moreover, in event-driven optimization mechanisms this problem results in very slow convergence to an optimum or in an algorithm failing to generate any improvement in the decision variables being updated.

In this work, we address this issue of event excitation in the context of multi-agent systems. In this case, the events required are often defined by an agent "visiting" a region or a single point in a mission space $S \subset \mathbb{R}^2$. Clearly, it is possible that such events never occur for a large number of feasible agent trajectories. This is a serious problem in trajectory planning and optimization tasks which are common in multi-agent systems seeking to optimize different objectives associated with these tasks, including coverage, persistent monitoring or formation control (Schwager et al., 2009; Cassandras et al., 2013; Cao et al., 2011; Oh and Ahn, 2014; Yamaguchi and Arai, 1994; Desai et al., 1999; Ji and Egerstedt, 2007; Wang and Xin, 2013). At the heart of this problem is the fact that objective functions for such tasks rely on a non-zero reward (or cost) metric associated with a subset $S^+ \subset S$ of points, while all other points in S have a reward (or cost) which is zero since they are not "points of interest" in the mission space. We propose a novel metric which allows *all* points in S to acquire generally non-zero reward (or cost), thus ensuring that all events are ultimately excited. This leads to a new method allowing us to apply event-based control and optimization to a large class of multi-agent problems. We will illustrate the use of this method by considering a general trajectory optimization problem in which Infinitesimal Perturbation Analysis (IPA) (Cassandras and Lafortune, 2006) is used as an event-driven gradient estimation method to seek optimal trajectories for

a class of multi-agent problems where the agents must cooperatively visit a set of target points to collect associated rewards (e.g., to collect data that are buffered at these points.) This defines a family within the class of Traveling Salesman Problems (TSPs) (Applegate et al., 2011) for which most solutions are based on techniques typically seeking a shortest path in the underlying graph. These methods have several drawbacks: (*i*) they are generally combinatorially complex, (*ii*) they treat agents as particles (hence, not accounting for limitations in motion dynamics which should not, for instance, allow an agent to form a trajectory consisting of straight lines), and (*iii*) they become computationally infeasible as on-line methods in the presence of stochastic effects such as random target rewards or failing agents. As an alternative we seek solutions in terms of parameterized agent trajectories which can be adjusted on line as a result of random effects and which are scalable, hence computationally efficient, especially in problems with large numbers of targets and/or agents. This approach was successfully used in (Lin and Cassandras, 2015), (Khazaeni and Cassandras, 2015).

1.5.3 Event-driven Trajectory Optimization in Multi-agent Systems

In the data harvesting problem the task is not completed by collecting the data from the data generating points. Data should be collected and delivered to the sink node or the base for the task to be completed. The main goal is to collect and deliver the data in the most efficient way. The general optimization problem is formulated as an optimal control problem. We aim to optimize a two-dimensional trajectory for each agent, which may be periodic and can collect data from a target once the agent is within a given range from that target.

The system at hand is a stochastic hybrid system with discrete modes defined depending on the dynamics of the agent and targets. We note that the specification of an appropriate objective function is nontrivial for the data harvesting problem,

largely due to the fact that the agents act as "mobile servers" for the data sources and have their own dynamics. Since the control is applied to the motion of agents, the objective function must capture the agent behavior in addition to that of the data queues at the targets, the agents, and the base. The solution of this optimal control problem (even in the deterministic case) requires a Two Point Boundary Value Problem (TPBVP) numerical solver which is clearly not suited for on-line operation and yields only locally optimal solutions. Thus, the main contribution of this part is to formulate and solve an optimal parametric agent trajectory problem. In particular, we represent an agent trajectory in terms of general function families characterized by a set of parameters that we seek to optimize, given an objective function. We consider elliptical trajectories as well as the much richer set of Fourier series trajectory representations. We demonstrate the application of Infinitesimal Perturbation Analysis for this hybrid system (Cassandras et al., 2010) to estimate gradients of the objective function with respect to the trajectory parameters and subsequently obtain (at least locally) optimal trajectories.

This approach also allows us to exploit (*i*) robustness properties of IPA to allow stochastic data generation processes, (*ii*) the event-driven nature of the IPA gradient estimation process which is scalable in the event set of the underlying hybrid dynamic system, and (*iii*) the on-line computation which implies that trajectories adjust as operating conditions change (e.g., new targets).

1.6 Outline of This Thesis

In chapter 2 we will formally introduce MRCP problem with detailed mathematical formulation. A brief review of the previous CRH controller is given with explanations of some of its shortcomings. After that, the new CRH controller is introduced along with all the elements of the new design. The chapter ends with several simulation

results with comparisons presented with the previous CRH controller and also with TSP solutions in case of one agent scenarios.

In chapter 3 we propose a general framework for static and dynamic multi-agent systems. We focus our interest on systems with finite number of point of interests studying the application of event-driven optimization methods in such systems. We show that in some cases due to no event excitation , the application of event-driven methods to such problems will not be straight forward. We then move to introduce a new metric to address this issue. Some simulation examples are also provided in this chapter for a data collection problem.

In chapter 4 we apply the event-driven optimization method to a parametric trajectory optimization for data harvesting problem.

In chapter 5 conclusions of the current work is presented continued with some future direction for possible extensions and several new paths of work.

Chapter 2

Maximum Reward Collection Problem

2.1 Problem Description

The Maximum Reward Collection Problem (MRCP) is a dynamic optimization problem which consists of multiple agents collecting a set of fully/partially known targets with unequal time-dependent rewards all located in a compact space S. The final objective is to maximize the total collected reward from all the targets. The available time for visiting these targets may be limited by a mission deadline, where not all the targets might be visited and the optimal solution should involve picking the optimal subset of targets to be visited. Alternatively it can be an open ended problem where all targets will be visited and the goal is to collect the most possible reward. Notice that considering an uncertain environment and also time-dependent rewards for the targets makes standard TSP solution algorithm not applicable to this problem. Even in the absence of any uncertainty, the TSP algorithm are not applicable to problems with non uniform time-dependent rewards.

We define a *mission* as collecting the maximum possible total reward from M number of targets by N number of agents in the minimum amount of time or by a pre-defined mission deadline T. Upon collecting rewards from all targets, the reward is then delivered to a common point defined as the *Base* or *Depot* whose location is denoted by $\mathbf{z} \in S$. If there is no target left, every agent heads back to the base.

Events happening during a MRCP mission can be controllable or random. Collecting a target when the agent is going towards it, is a predictable and controllable

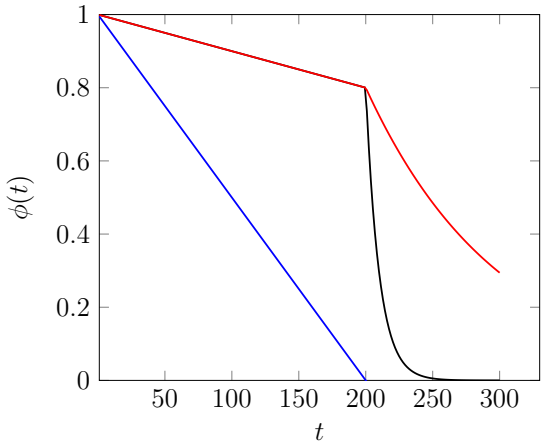


Figure 2·1: Black Curve: $D_i = 200, \alpha_i = 0.2, \beta = 0.1$ Red Curve:$D_i = 200, \alpha_i = 0.2, \beta = 0.01$ Blue Curve:$D_i = 200, \alpha_i = 1$

event. Random events are due to the uncertainties rooted in the dynamics of the agents, random appearance and disappearance of targets, changes in their location, obstacles and threats in the problem environment. The event based CRH controller handles these random events by re-solving the optimal control problem at the time of any event.

A finite set of M points $\mathcal{T} = \{1, ..., M\}$ indexed by i in the mission space denotes the targets or visiting points. Target i's reward is denoted by $\lambda_i \phi_i(t)$ where λ_i is the initial reward and $\phi_i(t) \in [0, 1]$ is a non-increasing discount function. The location of each target is denoted by $\mathbf{y}_i \in S$. The special case of $\phi_i(t) = 1$ creates targets with fixed rewards. By using the right discounting function we can incorporate any type of constraints such as hard or soft deadlines for collecting any of the targets. A more elaborate example of the discounting function which incorporates deadline for the rewards by introducing a linear and exponential decline of the reward of the target is shown below:

$$\phi_i(t) = \begin{cases} 1 - \frac{\alpha_i}{D_i} t & \text{if } t \leq D_i \\ (1 - \alpha_i) e^{-\beta(t - D_i)} & \text{if } t > D_i \end{cases} \tag{2·1}$$

The target's reward is declining linearly before the deadline D_i and exponentially after that. α_i is the linear decline factor for target i and β is a constant value that changes the rate of the exponential decline. Figure 2·1 shows three examples of the discounting function $\phi(t)$ where the deadline $D_i = 200$, The red and black curve has the same linear decline while the black one has a rapid exponential decline to model a hard deadline and the red curve shows a more gradual decline after the deadline D_i. The blue curve is a linearly decreasing reward with the same deadline $D_i = 200$ and no exponential decline.

There is N agents in the mission, $\mathcal{A} = \{1, ..., N\}$ indexed by j. Location of the agents is denoted by $\mathbf{x}_j(t) \in S$. Each agent has a constant speed of V_j and the controllable value for each agent at time t is its heading $u_j(t)$. The velocity of the agent is then defined using the heading $u_j(t)$ and it's speed V_j as

$$\mathbf{v}_j(t) = V_j \begin{bmatrix} \cos(u_j(t)) \\ \sin(u_j(t)) \end{bmatrix} \tag{2·2}$$

Each agent can head to a set of feasible directions $\mathbf{U}_j(t)$ at any location and time. We also assume each agent can change its heading simultaneously so the agent's dynamic properties are not a limiting issue in the problem.

Distance Metric

The distance metric $d : S \times S \to \mathbb{R}$ for each mission is defined based on the topology of S. In general, we denote the real valued distance metric $d(x, y)$ as the length of the shortest path between points x and $y \in S$. The distance metric that is defined for each topology is different and should inherit the properties of the mission space.

In order to ensure that the agents will eventually collect the targets in finite time, one assumption that we have to make is that each target has a capture radius or size. Once an agent is within a specified finite distance to a target it can collect

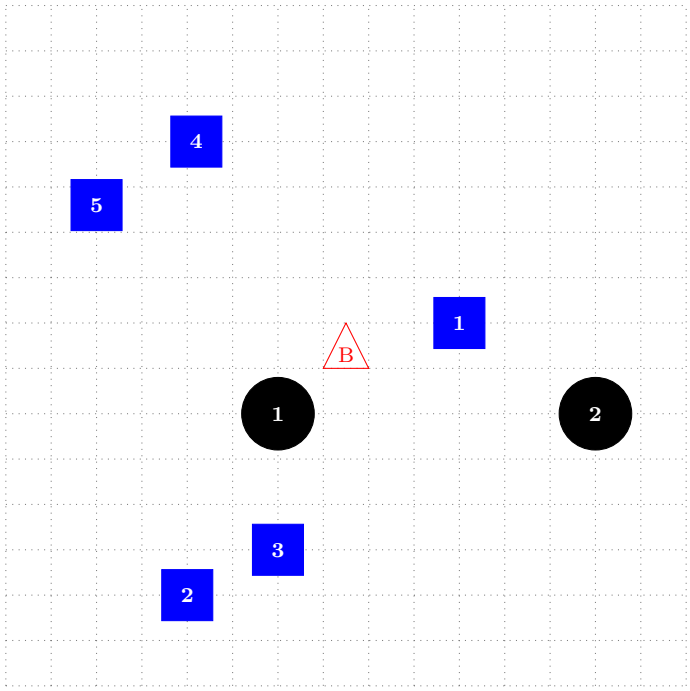


Figure 2·2: Sample mission space with 2 agents (Black circles) and 5 targets (Blue squares) and one base (Red triangle)

the reward from that target and move on with the rest of the mission. Assuming a size s_i for each target i, we define that agent j collects target i at time t if and only if $d(\mathbf{x}_j(t), \mathbf{y}_i) \leq s_i$.

2.1.1 Mission Space Topology

Euclidean Topology

Mission spaces can have different topologies within the MRCP work frame. In a Euclidean topology, the mission space S is a subset of $\mathbb{R}^2$. In Figure 2·2 a sample of the Euclidean mission space is shown where the red triangle in the middle shows the base or depot. The black circles show the agents 1 and 2. Five targets are shown with blue squares. Figure 2·3 shows a similar mission space where some parts of it are areas that contain obstacles where agents are not supposed to pass through.

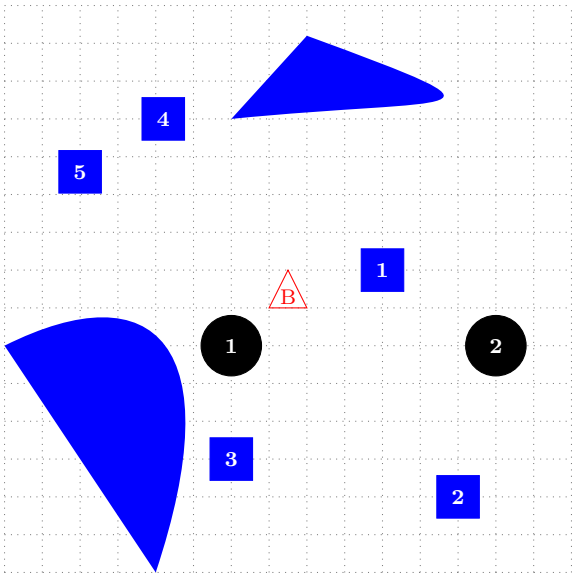


Figure 2·3: Sample mission space with filled blue regions as obstacles

In the Euclidean topology, the feasible headings $\mathbf{U}_j(t) = [0\ 2\pi], \quad \forall t$ for each agent with no obstacle in the mission space. If there is any obstacles that would block some of the directions. Also if there is no obstacles in S the distance metric $d(x, y)$ is a simple Euclidean distance while if there is any obstacles in the set S the possible shortest path that doesn't pass through the obstacles creates the distance function.

Graph Topology

In a graph topology mission, the mission space S is a (directed) graph $\mathcal{G}(E, V)$ where the set $E = \{1, ..., M\} + \{B\}$ denotes the graph nodes or the target set plus the base location, $\{B\}$. Agents can only move on the edges of the graph in the set V. The feasible set of headings for each agent at any point is defined by the available (directed) edges at that point. A special case of the graph topology is a grid where at each point a fixed set of headings is available to each agent. In a (directed) graph $\mathcal{G}(E, V)$ the distance $d(u, v)$ is defined as the sum of the weights on the (directed)

edges that build the shortest path between u and v.

2.2 An Event-driven Optimal Control View

The complete solution of MRCP is sequences of headings for all agents and synchronous heading switching times. We define a policy $\boldsymbol{\pi}$ as a vector $[\mathbf{u}, \xi]$ where $\xi = [\xi_1, \xi_2, \xi_3, ..., \xi_K]$ are the switching time intervals that headings are maintained for. The switching time $t_{k+1} = \sum_{l=1}^{k} \xi_l$, where t_1 is set to 0. The headings $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, ..., \mathbf{u}_K]$ where $\mathbf{u}_k = [u_1(t_k), ..., u_N(t_k)]'$ is the vector of all the agents' heading at time t_k. With the number of targets being bounded we can say with a finite number of switching, we visit all targets in the mission space. Each switching time t_k is either the result of a controllable event like visiting a target, or an uncontrollable event like an agent's failure or a new target appears. This is a complex stochastic control problem where the state space $\mathscr{X}$ is the set of all possible location of agents $\mathcal{X}_k = [\mathbf{x}_1(t_k), ..., \mathbf{x}_N(t_k)]$ and targets $\mathcal{Y}_k = [\mathbf{y}_1, ..., \mathbf{y}_{M_k}]$. Assuming $\mathcal{T}_k$ is the set of unvisited targets at time t_k and $\|\mathcal{T}_k\| = M_k$, The complete state of the problem at time t_k is $(\mathcal{X}_k, \mathcal{Y}_k) \in \mathscr{X}$. With the policy $\boldsymbol{\pi}$ defined previously, we define the optimization problem $\mathbf{P}$ as:

$$\mathbf{P}: \qquad \max_{\boldsymbol{\pi}} \sum_{k=1}^{K} R_{\boldsymbol{\pi}}(t_k, \mathcal{X}_k, \mathcal{Y}_k) \tag{2·3}$$

where

$$R_{\boldsymbol{\pi}}(t_k, \mathcal{X}_k, \mathcal{Y}_k) = \sum_{i=1}^{M_k} \sum_{j=1}^{N} \lambda_i \phi_i(t_k) \mathbb{1}\{d(\mathbf{x}_j(t_k), \mathbf{y}_i) \leq s_i\} \tag{2·4}$$

Defining any time a target is visited as a controllable event, any visiting time automatically is a switching time. In a completely deterministic problem there is no need to switch heading unless a target is visited but in an uncertain mission the switching times are not limited to these events. We define a subset of $\{t_1, ...t_K\}$ as $\boldsymbol{\tau} = [\tau_1, \tau_2, ..., \tau_M]$, $M \leq K$ so that τ_i denotes the time target i is collected. $\boldsymbol{\tau}$ is

not a monotonic sequence and targets can be picked up at any order. $\sigma_{\boldsymbol{\pi}}$ is defined as a permutation of $\{1, 2, .., M\}$ to be the order in which the targets are collected under policy $\boldsymbol{\pi}$. We can sort the vector $\boldsymbol{\tau}$ into $[\tau_{\sigma_{\boldsymbol{\pi}}(1)}, \tau_{\sigma_{\boldsymbol{\pi}}(2)}, ..., \tau_{\sigma_{\boldsymbol{\pi}}(M)}]$ as the ordered switching times at which one target is collected. Now another formulation of (2·3) is:

$$\max_{\boldsymbol{\pi}} \sum_{i=1}^{M} \lambda_i \phi_i(\tau_i(\sigma_{\boldsymbol{\pi}}(i))) \tag{2·5}$$

Solving problem $\mathbf{P}$ for the complete policy $\boldsymbol{\pi}$ is a complex stochastic control problem. Instead let's solve an optimal control for one step of the policy $(\mathbf{u}_k, \xi_k)$. Defining the immediate reward as the reward collected during ξ_k period of time, and the rest of the reward as an aggregated over $t > t_k + \xi_k$. The optimality equation of this problem is:

$$J^*(t_k, \mathcal{X}_k, \mathcal{Y}_k) = \max_{\mathbf{u}_k, \xi_k} (J_I(t_k, \mathcal{X}_k, \mathcal{Y}_k, \mathbf{u}_k, \xi_k) + J^*(t_{k+1}, \mathcal{X}_{k+1}, \mathcal{Y}_{k+1})) \tag{2·6}$$

where $J^*(t_k, \mathcal{X}_k, \mathcal{Y}_k)$ denotes the maximum possible total reward to be collected at time t_k with $(\mathcal{X}_k, \mathcal{Y}_k)$ be the current state of the problem and $J_I(t_k, \mathcal{X}_k, \mathcal{Y}_k, \mathbf{u}_k, \xi_k)$ is the immediate reward collected in the interval of $(t_k, t_{k+1}]$ with length ξ_k. Hereunder, we for brevity, we are dropping the $\mathcal{X}_k$ and $\mathcal{Y}_k$ from the arguments list.

Had we known the switching times ξ_k *a priori*, the optimization problem could be possibly solved using dynamic programming(DP) method starting from a terminal state. The terminal state are when there is no target left in the mission space. But having the switching intervals ξ as part of the control parameters is the main issue in solving (2·6) using DP. This issue and also the size of the state space of the problem, proves DP to be an impractical solution method for this problem. We instead switch to a forward moving sequential optimization scheme based on the receding horizon control method.

We assume at time step t_k we are given a planning horizon H_k. This planning horizon allows us to calculate the next switching time $t_{k+1} = t_k + H_k$ and define

an optimal control problem for the interval $(t_k, t_k + H_k]$. The finite horizon optimal control problem is solved to find the control $\mathbf{u}_k$. Maintaining this control for an action horizon h_k, a new optimization problem is re-solved at $t_{k+1} = t_k + h_k$ or earlier if any random event happens. We define the optimization problem $\mathbf{P}_k$ as:

$$\mathbf{P}_k : \qquad J^*(t_k, H_k) = \max_{\mathbf{u}_k}(J_I(\mathbf{u}_k, t_k, H_k) + J^*(t_{k+1}, H_{k+1})) \tag{2·7}$$

$J^*(t_k, H_k)$ denotes the maximum total reward that is possible to be collected when we are at time t_k and $J_I(\mathbf{u}_k, t_k, H_k)$ is the immediate reward that can be collected in the interval of $(t_k, t_k + H_k]$. This immediate reward is zero if the agents don't visit any target after H_k otherwise it is the reward of as many targets that are collected in $(t_k, t_k + H_k]$. Obviously $J^*(t_{k+1}, H_{k+1})$ is also the maximum total reward that is possible to be collected at time t_{k+1}. Notice that here the maximization is only on $\mathbf{u}_k$ compared to the original problem where the total reward was maximized on the complete control policy.

2.3 Review of The Previous CRH Controller

In this section we discuss the previous CRH controller introduced in (Li and Cassandras, 2006b) and related works. We bring up the limitations of this approach and will present the methods for handling them in the next sections. Here we review the cooperation scheme and planning horizon calculation that will also be used in this work.

2.3.1 Cooperation Scheme

In (Li and Cassandras, 2006b), the agents divide the mission space into a dynamic partition at each step of the mission. The CRH controller does not assign a target to an agent in any step. All agents are responsible for all targets but the degree of this

responsibility depends on the relative proximity of that agent to each target.

Depending on the cooperation level, A neighbor set is defined for each target point which includes the b closest agents to that target. These agents will be the only ones that are responsible for that target until another agent moves closer in the future. Assuming $b = 2$, then at each time only two of the agents will share the responsibility of any target, obviously these will be the two closest agents to that target.

Defining $c_{ij}(t) = d(\mathbf{y}_i, \mathbf{x}_j(t))$ be the direct distance between target i and agent j at time t, let $B^l(\mathbf{y}_i, t)$ be the l^{th} closest agent to target i at time t. Formally,

$$B^l(i,t) = \underset{j\in\mathcal{A}, j\neq B^1(i,t),\ldots,j\neq B^{l-1}(i,t)}{\operatorname{argmin}} \{c_{ij}(t)\} \tag{2·8}$$

So the neighbor set is:

$$\beta^b(i,t) = \{B^1(i,t), ..., B^b(i,t)\}. \tag{2·9}$$

Based on this neighbor set a *relative distance* function is then defined for all agents in the mission :

$$\delta_{ij}(t) = \begin{cases} \dfrac{c_{ij}(t)}{\sum_{k\in\beta^b(i,t)} c_{ik}(t)} & \text{if } j \in \beta^b(i,t); \\ 1 & \text{else.} \end{cases} \tag{2·10}$$

A value of $b = 2$ is used in the previous and current work. As a direct result from the definition, if the agent is not one of the two closest agents to one target assuming $b = 2$ then the relative distance is set to 1 and otherwise is less than 1.

The reward that will finally be collected from a target is viewed as an expected value of the reward given that it's collected by either of the agents in its neighbor set. To calculate this expected value a probability function is defined. This function measures the probability of the target being collected by a particular agent.

Probability function $p(\delta_{ij}(t))$ is defined as below and named *relative proximity func-*

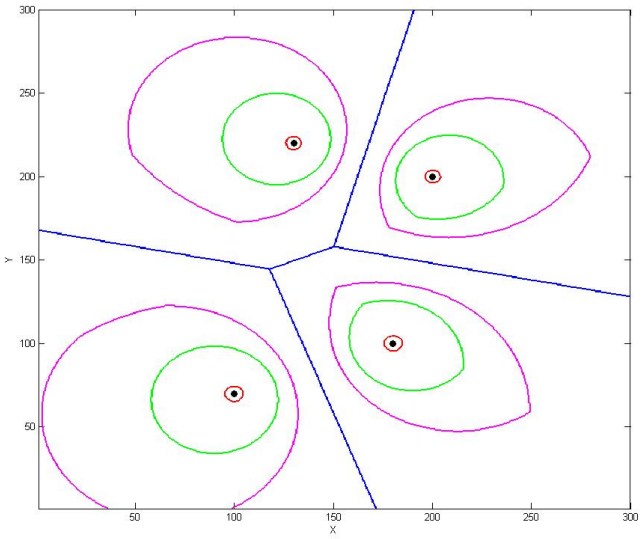


Figure 2·4: Cooperative partitions for 4 agents, location shown with black dots - $\Delta = 0.5$: Blue - $\Delta = 0.35$: Magenta - $\Delta = 0.25$: Green - $\Delta = 0.05$: Red

tion in (Li and Cassandras, 2006b).

$$p(\delta_{ij}(t)) = \begin{cases} 1, & \text{if } \delta \leq \Delta \\ \frac{1-\Delta-\delta}{1-2\Delta}, & \text{if } \Delta \leq \delta \leq 1-\Delta \\ 0, & \text{if } \delta > 1-\Delta \end{cases} \tag{2·11}$$

Here, $\Delta \in [0, \frac{1}{2})$ defines the level of cooperation between the agents. It can also be viewed as a "capture radius" around each agent. By increasing the Δ from 0 to $\frac{1}{2}$ the agents will take full responsibility for more targets and we generate less cooperation. In other words each agent takes on full responsibility for target i if $\delta_{ij}(t) \leq \Delta$. On the other hand, when $\Delta = 0$ no matter how close an agent is to a target, the two agents are still responsible for that target. If $\Delta = \frac{1}{2}$ then the closest agent is always responsible and no cooperation happens between the two agents. In Figure 2·4 the cooperation regions and capture radius are shown for different values of Δ, (Li and Cassandras, 2006b). In this figure as an example, the green curves correspond to the $\Delta = 0.25$. For each of the four agents, if a target is inside their corresponding green

curve the value of $p(\delta_{ij}(t)) = 1$ meaning that agent is fully responsible for that target and no other agent would see that target. The region outside the green curves is the cooperation region. If a targets is in the cooperation region, for the two closest agents we have $0 < p(\delta_{ij}(t)) < 1$. This explanation applies to the red and magenta curves but the cooperation region is larger and smaller respectively. It can be noted that in the case of $\Delta = \frac{1}{2}$ the regions show the Voronoi tessellation (Okabe et al., 1992) of the mission space with the location of the agents to be the center of the Voronoi tiles. There is no cooperation region in this case and each agent is fully responsible for the targets within their own Voronoi tile.

An intuitive way to think about this probability function is that if a target is much closer to one of its neighboring agents than the second one, it will be collected by that agent with a higher probability. However if the agents are equally distanced from the target the fate of the system will be determined by other targets that are neighbor to those agents. This ensures that there is no target to agent assignment in this method. In fact the decision on which agent visits each target is left until the very last moment and can change if a new event happens in the mission space. This fact helps in the stochastic versions of the problem where targets can die or new targets can appear at any time and the location of the targets are not known prior to the beginning of the mission.

2.3.2 Planning and Action Horizons

In (Li and Cassandras, 2006b) H_k is defined s the shortest time until the first controllable event happens in the mission. This is the earliest time that one of the agents can potentially visit one of the targets, equation (2·12). This definition of *planning horizon* for the CRH controller ensures no controllable event can happen during this

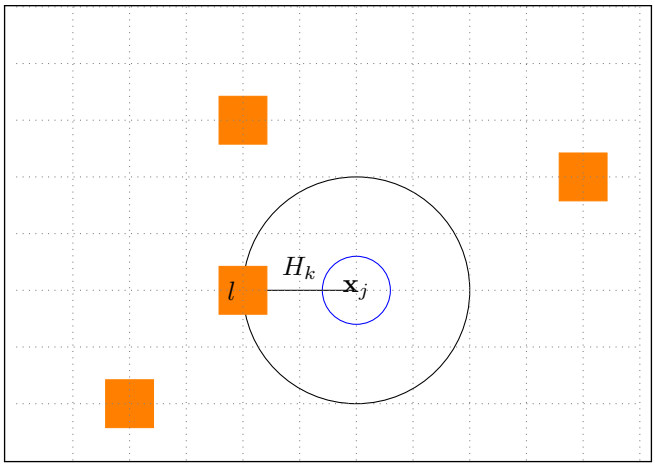


Figure 2·5: Calculation of Planning Horizon H_k

horizon. The process is shown in Figure 2·5.

$$H_k = \min_{l \in \mathcal{T}_k} \{ \frac{d(\mathbf{x}_j(t_k), \mathbf{y}_l)}{V_j} \} \tag{2·12}$$

The CRH control calculated at t_k is maintained for an action horizon $h_k \leq H_k$. In (Li and Cassandras, 2006b) h_k is defined with two factors. If a random event happens at $t_e \in (t_k, t_k + H_k]$ then $h_k = t_e - t_k$ Otherwise $h_k = \frac{H_k}{2}$ unless $H_k \leq r$ where $h_k = H_k$. Here r is a small threshold to discourage extremely small action horizons.

2.3.3 Limitations of the Previous CRH controller

The controller in the previous works (Li and Cassandras, 2006b) and (Li and Cassandras, 2003) has some limitations that are addressed in this work with new modifications.

Instabilities in Agent's Trajectory:

The objective function in (Li and Cassandras, 2006b) is modeled as a potential function, minimized in order to maximize the total reward. It is assumed all minima are at the targets locations (Condition C in (Li and Cassandras, 2006b)). If this assump-

tion is not holding the agent's are pushed toward the weighted center of gravity of all the targets. This can happen specifically in missions with some sort of symmetry, leading to oscillatory behavior of the agents. As an example in Figure 2.14(a) the previous CRH controller with one agent is shown while the agent oscillates between three targets with equal rewards. The agent goes toward target 3 but is attracted to the center of gravity of the three targets. This point does not fall on a target locations and results in instability. The problem was addressed in (Li and Cassandras, 2006a) by introducing a heading change cost factor C. This can prevent some of the instabilities but there is no guarantee that it helps the optimality of the results. Also the parameter C has to be tuned accordingly for each mission. A new method for calculating the expected reward is introduced to handle this issue.

Hedging and Mission Time:

The agent's trajectories in (Li and Cassandras, 2006b) tend to move the agents in positions close to targets but not exactly towards them. This hedging effect is helpful in handling uncertainties but it can create excessive loss of time specially when rewards are declining fast. This can be addressed by more direct movements toward the target but re-evaluating the control frequent enough that uncertainties in the mission space can be handled. The feasible control set in previous CRH is a continuous set, by reducing this continuous set to a discrete set of control values we can eliminate unnecessary hedging. This also reduces the complexity of the optimal control problem at each time step and allows for solving the problem by a finite number of evaluations.

Estimation of the Expected Reward:

In the previous CRH method introduced in (Li and Cassandras, 2006b), the estimated collection times are assumed to be the earliest time agent j would reach target i, given the control is $\mathbf{u}_k$ at time t_k and is maintained for H_k. $\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)$ is calculated as

an estimated collection time $\forall l \in \mathcal{T}_k$:

$$\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k) = t_k + H_k + d(\mathbf{x}_j(t_k + H_k, u_j(t_k)), \mathbf{y}_l) \tag{2·13}$$

There is no way the agents can visit all the targets at these collection times, therefore (2·13) is a lower bound estimation resulting in a loose upper bound estimation of the expected total reward. This estimation method is improved by a more realistic projection of agent's future moves.

2.4 The New CRH Controller

In this section a modified version of the CRH controller introduced in (Li and Cassandras, 2006b) is presented. In problem $\mathbf{P}_k$ in (2·7), at any time step t_k, we define the position of the agent in the next time step denoted by $\mathbf{x}_j(t_k + H_k, u_j(t_k))$ as a function of t_k, H_k and $\mathbf{u}_k$. Assuming that $V_j = 1$ for all agents, the feasible set for $\mathbf{x}_j(t_k + H_k, u_j(t_k))$ is defined as:

$$\mathcal{F}_j(t_k, H_k) = \{\mathbf{w} \in S |\ d(\mathbf{w}, \mathbf{x}_j(t_k)) = H_k\} \tag{2·14}$$

In a Euclidean mission $\mathcal{F}_j(t_k, H_k)$ is the circle centered at $\mathbf{x}_j(t_k)$ with radius H_k. Let the binary function $q_i(\mathbf{x}_j(t)) = \mathbb{1}\{d(\mathbf{x}_j(t), \mathbf{y}_i) \leq s_i\}$ show if agent j visits target i at time t. We define the immediate reward at t_k:

$$J_{\mathbf{I}}(\mathbf{u}_k, t_k, H_k) = \sum_{j=1}^{N} \sum_{l=1}^{M_k} \lambda_l \phi_l(t_k + H_k) q_l(\mathbf{x}_j(t_k + H_k, u_j(t_k))) \tag{2·15}$$

Following the definition of τ_i as the collection time of target i in (2·5), we define $\tilde{\tau}_{ij}$ as the estimated collection time of target i by agent j. Notice that here $\tilde{\tau}_{ij} > t_k$ is an estimated time so any of the agents in the mission space has a chance to visit target

i. Then at time t_k we can formulate an estimation of the $J^*(t_{k+1}, H_{k+1})$ as below:

$$\tilde{J}(\mathbf{u}_k, t_{k+1}, H_{k+1}) = \sum_{j=1}^{N} \sum_{l=1}^{M_{k+1}} \lambda_l \phi_l(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)) q_l(\mathbf{x}_j(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k))) \qquad (2{\cdot}16)$$

We previously mentioned that the previous approach used a lower bound for estimating $\tilde{\tau}_{ij}$. We try to improve this estimation by introducing a new parameter called *travel cost* for each target. This parameter combines the distance and reward of a target with a local sparsity factor. Using the travel cost factor, we introduce the active target set. The active targets definition allows us to shrink the infinite dimension feasible control set at each time step to a discrete set of controls. Finally we introduce the look ahead and aggregate algorithm with single and multiple steps.

2.4.1 Travel Cost Factor:

We define parameter $\zeta_i(t_k)$ for each target i measuring the sparsity of rewards around target i. Assuming that the upper-bounded deadline D_i shows the time the reward is zero, the average reward's rate of decay is then equal to λ_i/D_i. Let set $\{1, 2, ..., K\}$ be the indices for K closest targets to target i at time t_k. We define $\zeta_i(t_k)$ as a sparsity indicator around target i as:

$$\zeta_i(t_k) = \sum_{l=1}^{K} \gamma^l \frac{d(\mathbf{y}_i, \mathbf{y}_l)}{\lambda_l / D_l} \qquad (2{\cdot}17)$$

$\zeta_i(t_k)$ is a function of time because the K closest targets change during time while targets are collected. A larger $\zeta_i(t_k)$ shows target i is located in a sparse area and vice versa. $\gamma \in [0\ 1]$ is a parameter used to shift the weight between the K targets. K is chosen based on the number of targets in the mission space and the computation capacity of the controller. The main idea of this parameter comes from (Schneider et al., 2010) where it is used to solve TSP problems with clustering. For any point in

$\mathbf{x} \in S$, we define a travel cost to target i at time t_k :

$$\eta_i(\mathbf{x}, t_k) = \frac{d(\mathbf{x}, \mathbf{y}_i)}{\lambda_i / D_i} + \zeta_i(t_k) \tag{2·18}$$

The travel cost has a direct relationship with the distance, so the farther the target the more costly is the collection of that target. Also the reverse relationship with the target's rate of decay implies that the faster a target's reward is decaying, the less the travel cost is. $\zeta_i(t_k)$ is added to this cost so if a target is in a sparse area the travel cost of that target is higher.

2.4.2 Active Targets

We define a subset of the targets to be the possible next stops for each agent at time t_k. This set only defines the candidates for the next collection the agent won't necessarily visits them in the next step.

$$S_j(t_k, H_k) = \{\ell | \exists\ \mathbf{x} \in \mathcal{F}_j(t_k, H_k) \text{ s.t. } \ell = \underset{i \in \mathcal{T}_k}{\operatorname{argmin}}\, \eta_i(\mathbf{x}, t_k + H_k),\ i = 1, 2, ..., M_k\} \tag{2·19}$$

This set of targets is called the *Active Target Set* and the definition in (2·19) implies that a target is an active target if and only if it has the smallest travel cost from at least one of the points on the $\mathcal{F}_j(t_k, H_k)$. This means each point in the set $\mathcal{F}_j(t_k, H_k)$ corresponds to one of the active targets and so does each feasible heading. Heading $u_j(t_k)$ corresponds to active target l if and only if:

$$l = \underset{i \in \mathcal{T}_k}{\operatorname{argmin}}\, \eta_i(\mathbf{x}(t_k + H_k, u_j(t_k)), t_k + H_k) \tag{2·20}$$

Assuming that the distance metric $d(x, y)$ is a continuous function, The correspondence between the active targets and the feasible points set results in partitioning of the set $\mathcal{F}_j(t_k, H_k)$ into several arcs where each arc corresponds to one of the active targets. The common feature of all the points in one arc is that they correspond to

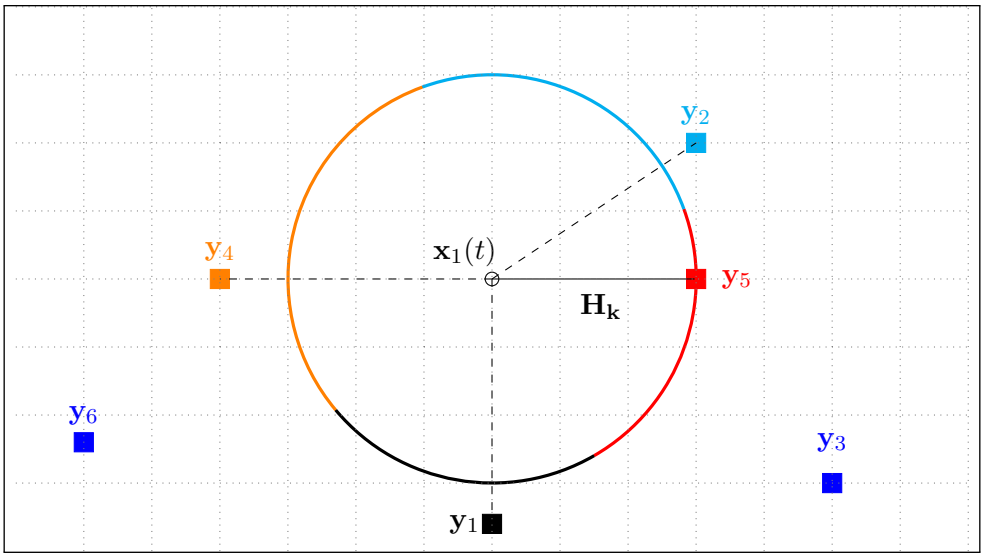


Figure 2·6: The Active Target Set for agent 1: $S_1(\mathbf{x}_1(t_k), H_k) = \{1, 2, 4, 5\}$

the same target with the least travel cost.

In Figure 2·6 an instance of the problem is illustrated. To be able to show the active targets we assume $\gamma = 0$ in (2·18) and the target have the same λ_i and $\phi_i(t)$. These assumptions enables us to reduce the travel cost factor to a simple Euclidean distance. Note that these assumptions are only for the illustration purposes in Figure 2·6 and are not carried over in the rest of the analysis.

In this simplified case, agent "1" has four active targets in its active target set, $S_1(t_k, H_k) = \{1, 2, 7, 8\}$. The feasible set is then divided into four partitions (arcs), each is associated with one of the active targets.

Active Target Set Construction

For each target l in the set $\mathcal{T}_k$ and each agent j we define the set $\mathcal{L}_k(\mathbf{x}_j(t_k), l)$ to be the set of points $\mathbf{x} \in S$ that defines the shortest path from $\mathbf{x}_j(t_k)$ to $\mathbf{y}_l$. In a Euclidean mission space we can define this set as a convex combination of $\mathbf{x}_j(t_k)$ and $\mathbf{y}_l$.

$$\mathcal{L}_k(\mathbf{x}_j(t_k), l) = \Big\{\mathbf{x} \in S | \mathbf{x} = (1-m)\mathbf{x}_j(t_k) + m\mathbf{y}_l; m \in [0\ 1]\Big\} \qquad (2\cdot21)$$

The intersection of these two sets denotes the set of closest points to target l in the feasible set $\mathcal{F}_j(t_k, H_k)$:

$$\mathcal{C}_{l,j}(t_k, H_k) = \mathcal{L}_k(\mathbf{x}_j(t_k), l) \cap \mathcal{F}_j(t_k, H_k) \tag{2·22}$$

Here $\mathcal{C}$ stands for crossing point since in a Euclidean mission this set is a single point where the circle $\mathcal{F}_j(t_k, H_k)$ and the line segment $\mathcal{L}_k(\mathbf{x}_j(t_k), l)$ cross each other.

Lemma 2.1. *Target l is an active target for agent j at time t_k if and only if*

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) \leq \eta_i(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k), \ \forall i \in \mathcal{T}_k \tag{2·23}$$

Proof. See Appendix. □

2.4.3 Action Horizon

h_k in (Li and Cassandras, 2006b) is defined either (i) through a random event that may be observed at $t_e \in (t_k, t_k + H_k]$ so that $h_k = t_e - t_k$, or (ii) as $h_k = \gamma H_k$, $\gamma \in (0, 1)$. This definition requires frequent iterations of the optimization problem through which $\mathbf{u}_k^*$ is determined in case no random event is observed to justify such action. Instead, when there are no random events, we define a new *multiple-immediate-target event* to occur when the minimization in (2·12) returns more than one target, i.e., the agent is at an equal distance from at least two targets. This is illustrated in Figure 2·7 where the agent is moving toward target 1 and at point $\mathbf{z}$ it is equidistant to targets 1 and 5. In this case, we define $h_k = \|\mathbf{z} - \mathbf{x}_1(t_k)\|$ and problem is re-solved at $t_k + h_k$. In general, we define h_k to be the shortest time until the first multiple-immediate-target event occurs in $(t_k, t_k + H_k]$:

$$\begin{aligned} h_k = \min \Big\{ H_k, \inf \{ t > t_k : \exists l, l^* \in \mathcal{T}_k \text{ s.t.} \\ d(\mathbf{x}_j(t_k + t, u_j(t_k)), \mathbf{y}_l) = d(\mathbf{x}_j(t_k + t, u_j(t_k)), \mathbf{y}_{l^*}) \} \Big\} \end{aligned} \tag{2·24}$$

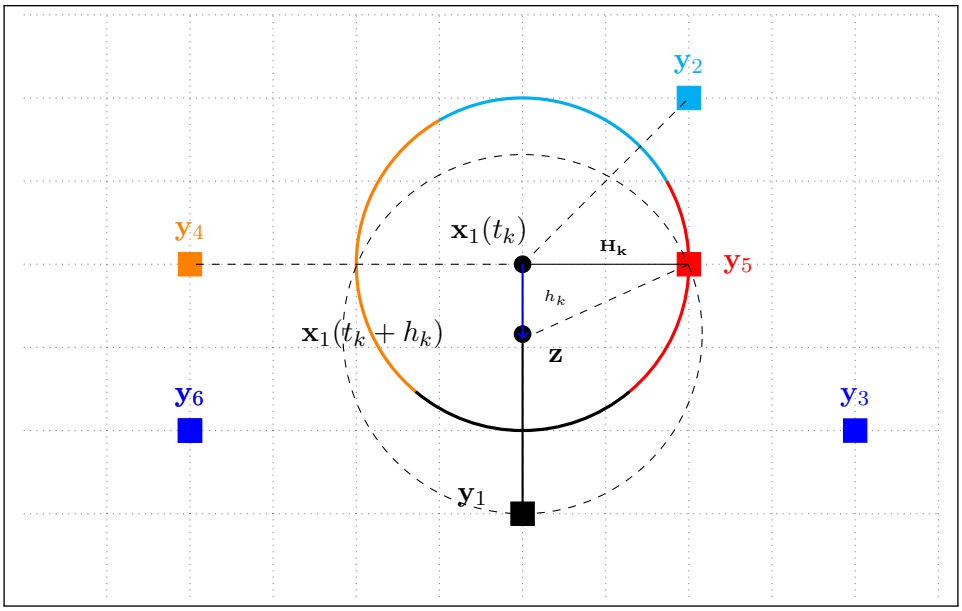


Figure 2·7: Multiple-Immediate-Target Event happens with agent at equal distance to targets 1 and 5

By (2·20), the change in $S_j(t_k, H_k)$ is in fact the earliest time when a CRH control re-evaluation is needed (unless an uncontrollable random event occurs) since the feasible control set remains otherwise unaffected. Consequently, this definition of h_k eliminates any unnecessary control re-evaluation. Although we use the same definition for the planning horizon H_k as in (2·12), we change the definition of the action horizon h_k in order to discourage un-necessary re-evaluations of the control. The definition of h_k in the previous approach is naively increasing the number of times CRH controller should re-solve a problem. Instead we define a new event for determining h_k in case of no random events:

Definition 2.1. *Multiple Immediate Target Event: This event happens when the minimization in* (2·12) *returns more than one target.*

We define the h_k to be the shortest time until the first multiple immediate target event happens in $(t_k, t_k + H_k]$. This definition tries to capture new active target while it prevents unnecessary control re-evaluations.

2.4.4 Look Ahead and Aggregate Algorithm

In order to solve the optimization problem $\mathbf{P}_k$ in (2·7) we need to estimate the collection time $\tilde{\tau}_{ij}(\mathbf{u}_k, t_k, H_k)$ for each $\mathbf{u}_k$. This is calculated using a projected path of the agent. The path projection has a *Look ahead* and *Aggregate* steps. In the first step the active target set is determined for each agent. With multiple agents in a mission, at each time step the remaining targets are partitioned using the relative proximity function in (2·11). We denote the partition for agent j as $\mathcal{T}_{k,j}$ where:

$$l \in \mathcal{T}_{k,j} \iff p(\delta_{lj}(t_k)) \geq p(\delta_{lq}(t_k)) \; \forall q \in \mathcal{A} \tag{2·25}$$

We assume $|\mathcal{T}_{k,j}| = M_{k,j}$. All $\tilde{\tau}_{ij}(\mathbf{u}_k, t_k, H_k)$ are estimated as if the agent would visit targets in its own partition by visiting the one with the least travel cost first. We define the permutation $\boldsymbol{\theta}^j(\mathbf{u}_k, t_k, H_k)$ to be the order of the targets in agent's j tour. We drop $\mathbf{u}_k$, t_k and H_k denoting the tour by $\boldsymbol{\theta}^j$ for simplicity. $\boldsymbol{\theta}^j(i)$ denotes the i^{th} target in agent j's tour. $\forall l \in \mathcal{T}_{k,j}$ and with $t_{k+1} = t_k + H_k$:

$$\eta_{\boldsymbol{\theta}^j(1)}(\mathbf{x}_j(t_{k+1}, u_j(t_k)), t_{k+1}) \leq \eta_l(\mathbf{x}_j(t_{k+1}, u_j(t_k)), t_{k+1}) \tag{2·26}$$

and and with $n = 2, ..., M_{k,j} - 1$, $\forall l \in \mathcal{T}_{k,j} - \{\theta(1), ..., \theta(n)\}$

$$\eta_{\boldsymbol{\theta}^j(n+1)}(\mathbf{y}_{\boldsymbol{\theta}^j(n)}, t_{k+1}) \leq \eta_l(\mathbf{y}_{\boldsymbol{\theta}^j(n)}, \tilde{\tau}_{\boldsymbol{\theta}^j(n),j}(\mathbf{u}_k, t_k, H_k)) \tag{2·27}$$

where

$$\tilde{\tau}_{\boldsymbol{\theta}(n),j}(\mathbf{u}_k, t_k, H_k) = t_k + H_k + \sum_{i=1}^{n-1} d(\mathbf{y}_{\boldsymbol{\theta}(i)}, \mathbf{y}_{\boldsymbol{\theta}(i+1)}) \tag{2·28}$$

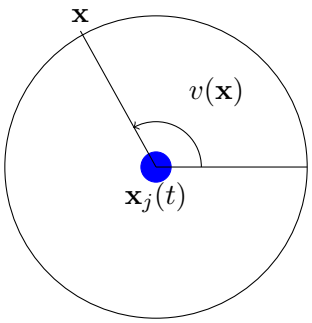


Figure 2·8: Agent's Heading in a Euclidean Mission Space

This results in the corresponding $\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)\ \forall l \in \mathcal{T}_{k,j}$. Now we calculate the reward-to-go assuming $|\mathcal{T}_{k,j}| = M_{k,j}$.

$$J_{\mathbf{A}}(\mathbf{u}_k, t_k, H_k) = \sum_{j=1}^{N} \sum_{l=1}^{M_{k+1,j}} \lambda_l \phi_l(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k)) q_l(\mathbf{x}_j(\tilde{\tau}_{lj}(\mathbf{u}_k, t_k, H_k))) \tag{2·29}$$

Bringing the immediate reward from (2·15), the optimization problem $\mathbf{P}_k$ is:

$$\max_{\mathbf{u}_k \in [0\ 2\pi]^N} [J_{\mathbf{I}}(\mathbf{u}_k, t_k, H_k) + J_{\mathbf{A}}(\mathbf{u}_k, t_k, H_k)] \tag{2·30}$$

In (2·14) we defined the feasible set for the location of agent j in the next step $t_{k+1} = t_k + H_k$. In a Euclidean mission space, each point $\mathbf{x} \in \mathcal{F}_j(t_k, H_k)$ corresponds to a heading $v(x)$ which is the angle shown in Figure 2·8. The value of the heading for each point $\mathbf{x}$ can be easily calculated by simple trigonometric functions. In lemma 2.2 we will prove that with the objective function defined as in (2·30) the optimal $u_j(t_k) = v(\mathcal{C}_{l,j}(t_k, H_k))$ for some $l \in S_j(t_k, H_k)$

Lemma 2.2. *In a single agent mission* ($j = 1$), *if* $\mathbf{u}^* = [u_1^*]$ *is an optimal solution to the problem:*

$$J^*(t_k, H_k) = \max_{\mathbf{u}_k \in [0\ 2\pi]} [J_{\mathbf{I}}(\mathbf{u}_k, t_k, H_k) + J_{\mathbf{A}}(\mathbf{u}_k, t_k, H_k)] \tag{2·31}$$

then

$$\exists\ l \in S_1(t_k, H_k) \quad s.t.\ u_1^* = v(\mathcal{C}_{l,1}(t_k, H_k)) \tag{2·32}$$

Proof. See Appendix. □

Now we can reduce the number of feasible controls to a countable set compared to the infinite set of $[0\ 2\pi]$. Let's define the set $\mathcal{V}_j$ as the new feasible headings for agent j:

$$\mathcal{V}_j(t_k, H_k) = \{v(\mathbf{x}) | \mathbf{x} = \mathcal{C}_{l,j}(t_k, H_k), l \in S_j(t_k, H_k)\} \tag{2·33}$$

Then the complete feasible control set is defined as:

$$\mathbf{V}_k = V_1(t_k, H_k) \times V_2(t_k, H_k) \times ... \times V_N(t_k, H_k) \tag{2·34}$$

Theorem 2.1. *In a multi-agent MRCP mission if* $\mathbf{u}^* = [u_1^*, ..., u_N^*]$ *is the optimal solution to the problem in* (2·30) *then* $\mathbf{u}^* \in \mathbf{V}_k$.

Proof. See the Appendix □

Theorem 2.1 reduces the problem $\mathbf{P}_k$ to a maximization problem over a countable set of feasible controls.

$$J^*(t_k, H_k) = \max_{\mathbf{u}_k \in \mathbf{V}_k} [J_{\mathbf{I}}(\mathbf{u}_k, t_k, H_k) + J_{\mathbf{A}}(\mathbf{u}_k, t_k, H_k)] \tag{2·35}$$

This reduces the size of the problem compared to the previous version of the CRH controller. An example is shown in Figure 2·9. Two feasible points $\mathbf{x}$ and $\mathcal{C}_{5,1}$ corresponding to two feasible heading toward the same arc are shown. These feasible headings have a common corresponding active target 5. For both feasible heading the projected path would be $[5, 4, 1, 3, 6, 2]$. Notice that for the case that the active target is positioned at one of the feasible points and may be collected after $t_k + H_k$ the same holds. This is also shown in Figure 2·9 for active target 6 as for any point on the red arc which corresponds to this active target, the projected path would be $[6, 2, 3, 1, 5, 4]$.

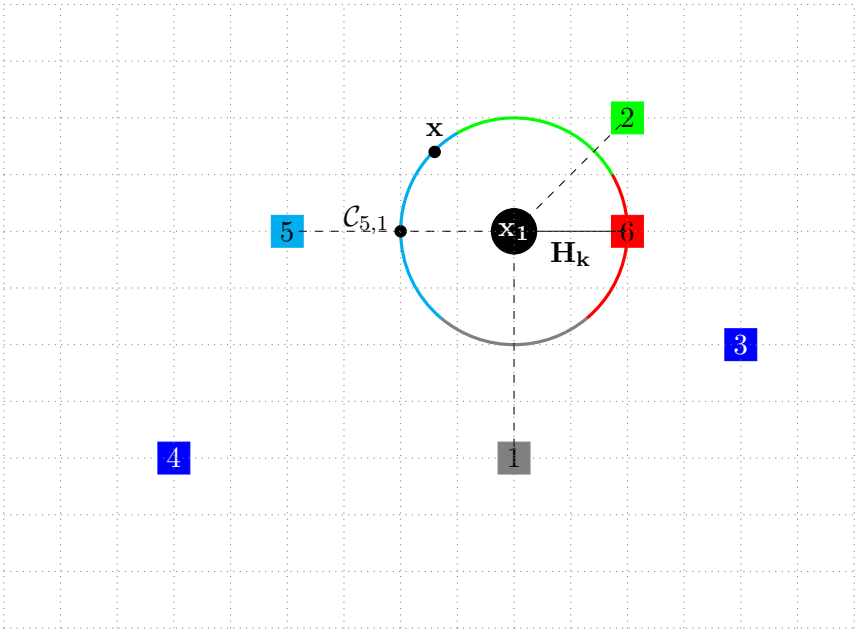


Figure 2·9: Two Different Feasible Points in the Set $\mathcal{F}_j(t_k, H_k)$

Multi-Step Look Ahead and Aggregate

Continuing with the same idea we can extend the One Step Look Ahead CRH to a Multiple Steps Look Ahead algorithm. The idea is to investigate more possible future paths for each agent at each time step t_k. In the One Step Look Ahead CRH, the aggregation reward is calculated based on one estimated tour of the remaining targets. However, as we remember in the original problem $\mathbf{P}_k$ in (2·7) the reward-to-go $J^*(t_{k+1}, H_{k+1})$ itself is the maximum reward that can be collected at time t_{k+1}. The Multiple Step Look Ahead algorithm tries to estimate this maximum reward by investigating more possible tours for each agent. For any feasible $u_j(t_k) \in \mathcal{V}_j(t_k, H_k)$ we can hypothetically move the agent to the next time step location $\mathbf{x}_j(t_{k+1})$. We do this for all the agents so we keep the synchronicity of the solution. At this new hypothetical position, a new set of active targets is determined for each agent. Now each agent can have $|S_j(t_k+H_k, H_{k+1})|$ number of possible paths. At this point, we can repeat the same procedure by hypothetically moving the agent to a feasible location

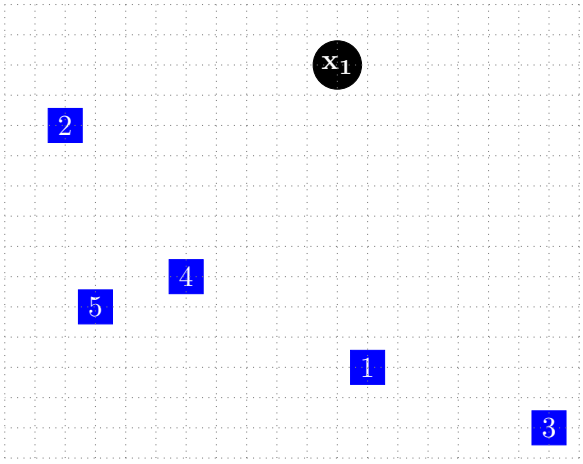


Figure 2·10: Sample mission with 5 targets and 1 agent

from the set $\mathcal{F}_j(t_{k+1}, H_{k+1})$ or we can stop and calculate an estimated reward for each available path. Once the estimated reward for each of these paths is calculated, the maximum possible reward is the new aggregated reward. For a Two Step Look Ahead problem $\mathbf{P}_k$ is:

$$\max_{\mathbf{u}_k \in \mathbf{V}_k} \left[J_{\mathbf{I}}(\mathbf{u}_k, t_k, H_k) + \max_{\mathbf{u}_{k+1} \in \mathbf{V}_{k+1}} \left[J_{\mathbf{I}}(\mathbf{u}_{k+1}, t_{k+1}, H_{k+1}) + J_{\mathbf{A}}(\mathbf{u}_{k+1}, t_{k+1}, H_{k+1}) \right] \right] \tag{2·36}$$

This procedure can easily be repeated as many time as needed. For a single agent case if we repeat this procedure at each time step until there is no target available. This means for each agent we exploit all possible paths that it can take through all the targets assuming at any t_k it can go toward one of its active targets. We then obtain a tree structure that the root shows the starting point or the base and a path from the root to each leaf is a possible path for the agent. In Figure 2·10 a sample mission with 5 targets is shown. If the optimal path of the agent is going to be calculated using a brute force method the total number of possible paths is $5! = 120$. The tree structure for this mission has 21 possible paths. The tree is shown in Figure 2·11. Note that to save the space the tree is broken into two parts . The first active target

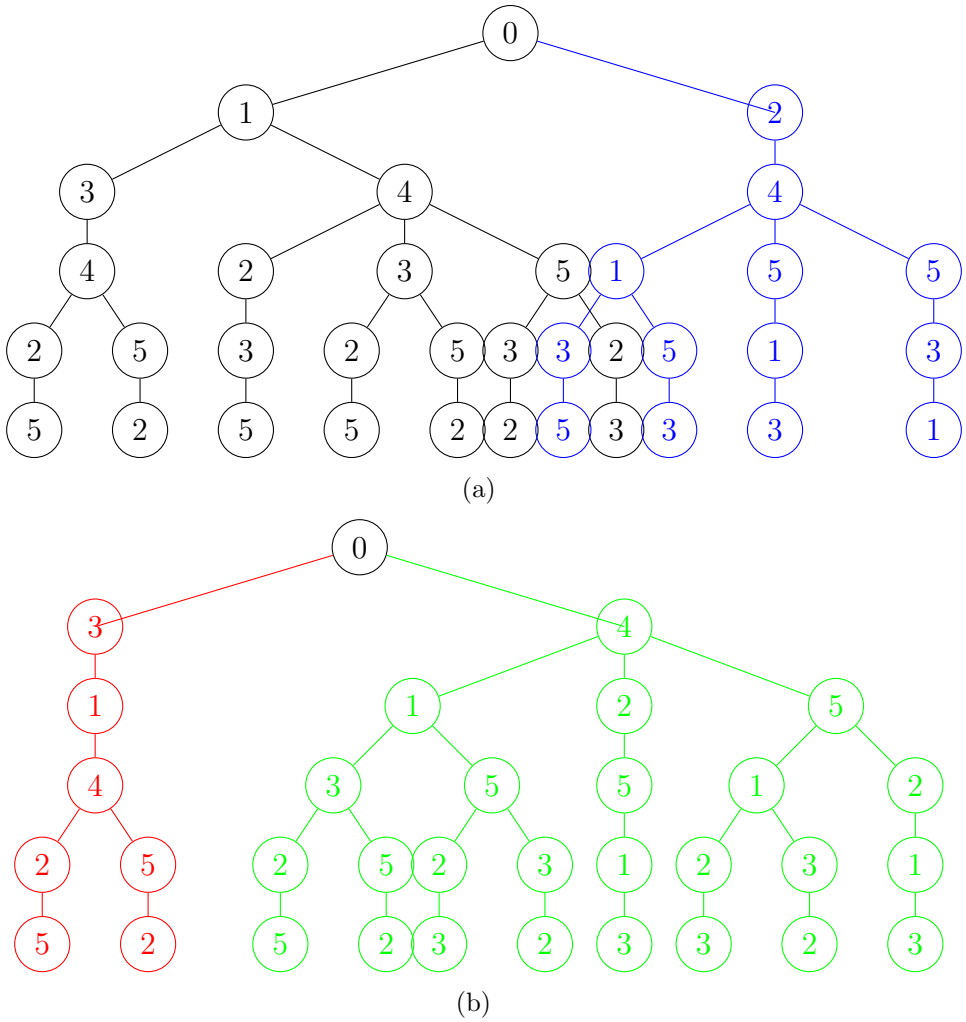


Figure 2·11: The tree structure for the 5 target mission

set for agent 1 consists of targets $1, 2, 3, 4$. Each of these active targets would then generate several branches in the tree. These branches are shown in different colors for more clarity. The starting point shown in point 0 is agent 1's initial location.

However, finding the complete tree for problems with more than a handful of targets is very time consuming. The Multi-Step Look Ahead CRH controller enables us to investigate the tree down to a few levels and then calculate an estimated reward for the rest of the branch. In order to have a better understanding of the Multi-Step Look Ahead algorithm, the procedure for two cases are presented as separate

algorithms. In algorithm 1 the Two-Step Look Ahead for a one agent case is shown. The number of agents is assumed to be one only for the simplicity of the notation in the algorithm. In the more complicated version of the algorithm, the K-Step Look Ahead for N agents is shown in algorithm 2. This algorithm has a recursive part that is repeated until the K step look ahead is performed.

Algorithm 1 Two Step Look Ahead and Aggregate for One Agent

$j = 1$ One Agent problem
Find Active Targets $S_1(t_k, H_k)$
Define $S_k = |S_1(t_k, H_k)|$
Define Set $\mathcal{V}_j(t_k, H_k)$ as in 2·33.
for $i = 1 : S_k$ **do**
    Next heading v_i for agent j : $v_i = i^{th}$ element of $\mathcal{V}_j(t_k, H_k)$
    Calculate $J_{\mathbf{I}}(v_i, t_k, H_k)$ (equation 2·15).
    Move the agent to $\mathbf{x}_j(t_k + H_k, v_i)$
    $t_{k+1} = t_k + H_k$
    Find H_{k+1}
    Find Active Targets $S_j(t_{k+1}, H_{k+1})$
    Define $S_{k+1} = |S_j(t_{k+1}, H_{k+1})|$
    Define Set $\mathcal{V}_j(t_{k+1}, H_{k+1})$ as in 2·33.
    for $ii = 1 : S_{k+1}$ **do**
        Next heading v_{ii} for agent j : $v_{ii} = ii^{th}$ element of $\mathcal{V}_j(t_{k+1}, H_{k+1})$
        Calculate the Next Position for Agent 1 : $\mathbf{x}_j(v_{ii}, t_k + H_k)$
        Calculate $J_{\mathbf{I}}(v_{ii}, t_{k+1}, H_{k+1})$ (2·15).
        Find the aggregation tour starting at $\mathbf{x}_j(t_k + H_k, v_{ii})$.
        Calculate $J_{\mathbf{A}}(v_{ii}, t_{k+1}, H_{k+1})$ (2·16).
    end for
    Calculate the Maximum Reward to Go:

$$J^*(v_i, t_{k+1}, H_{k+1}) = \max_{ii} \left[J_{\mathbf{I}}(v_{ii}, t_{k+1}, H_{k+1}) + J_{\mathbf{A}}(v_{ii}, t_{k+1}, H_{k+1}) \right]$$

    Calculate the Estimated Total Reward:

$$J(v_i, t_k, H_k) = J_{\mathbf{I}}(v_i, t_k, H_k) + J^*(v_i, t_{k+1}, H_{k+1})$$

end for
$J^*(t_k, H_k) = \max_i J(v_i, t_k, H_k)$
$\mathbf{u}_k^* = \text{argmax}_{v_i} J(v_i, t_k, H_k)$

Algorithm 2 K Step Look Ahead CRH for N Agents

1: Steps=K
2: **for** j=1:N **do**
3: Find Active Targets $S_j(t_k, H_k)$ and define $S_{k,j} = |S_j(t_k, H_k)|$
4: **end for**
5: Calculate $\mathbf{V}_k$ and the number of feasible controls: $T_k = |\mathbf{V}_k| = \prod_{j=1}^{N} |S_j(t_k, H_k)|$
6: **for** $i = 1 : T_k$ **do**
7: Take $\mathbf{v}_i = i^{th}$ row of $\mathbf{V}_k$
8: **for** $j = 1 : N$ **do**
9: Next heading for agent j : $v_i(j)$ and Next location: $\mathbf{x}_j(t_k + H_k, v_i(j))$
10: **end for**
11: Calculate $J_{\mathbf{I}}(\mathbf{v}_i, t_k, H_k)$ as in equation 2·15.
12: $t_{k+1} = t_k + H_k$
13: Find H_{k+1}
14: $J_{Step} = 0$
15: Steps=Steps-1
16: **for** $j = 1 : N$ **do**
17: Find Active Targets $S_j(t_{k+1}, H_{k+1})$
18: Define $S_{k+1,j} = |S_j(t_{k+1}, H_{k+1})|$
19: **end for**
20: Calculate $\mathbf{V}_{k+1}$ and $T_{k+1} = |\mathbf{V}_{k+1}| = \prod_{j=1}^{N} |S_j(t_{k+1}, H_{k+1})|$
21: **for** $ii = 1 : T_{k+1}$ **do**
22: Take $\mathbf{v}_{ii} = ii^{th}$ row of $\mathbf{V}_{k+1}$
23: **for** $j = 1 : N$ **do**
24: Next heading for agent j : $v_{ii}(j)$
25: **end for**
26: $t_{k+1} = t_k + H_k$
27: Find H_{k+1}
28: Calculate $J_{\mathbf{I}}(\mathbf{v}_{ii}, t_{k+1}, H_{k+1})$ (2·15).
29: $J_{Step} = J_{\mathbf{I}}(\mathbf{v}_{ii}, t_{k+1}, H_{k+1}) + J_{Step}$
30: **if** Steps> 0 **then**
31: Go To line 15 with $k = k + 1$
32: **end if**
33: Partition the Rest of the Targets.
34: Find the Aggregation Tour for All Agent's Partitions.
35: Calculate $J_{\mathbf{A}}(\mathbf{v}_{ii}, t_{k+1}, H_{k+1})$ (2·16).
36: **end for**
37: $J^*(t_{k+1}, H_{k+1}) = \max_{ii} [J_{step} + J_{\mathbf{A}}(\mathbf{v}_{ii}, t_{k+1}, H_{k+1}]$
38: $J(t_k, H_k, \mathbf{v}_i) = J_{\mathbf{I}}(t_k, H_k, \mathbf{u}_k^i) + J^*(t_{k+1}, H_{k+1})$
39: **end for**
40: $J^*(t_k, H_k) = \max_i J(t_k, H_k, \mathbf{v}_i)$
41: $\mathbf{u}_k^* = \text{argmax}_i J(t_k, H_k, \mathbf{v}_i)$

2.4.5 Two Target and One Agent Case

The easiest case of the maximum reward collection problem is the case with one agent and two targets. Obviously this is an easy analytical routing problem which the solution is among the two possible paths the agent can take. However, we here prove that the one step look ahead and aggregate algorithm will solve the problem optimally with any linearly decreasing rewards. Consider the case shown in Figure 2.12(a). There is two targets in the mission with initial reward and deadline of λ_1, D_1 and λ_2, D_2 respectively. The analytical solution for this case reveals wether path $1 \rightarrow 2$ or $2 \rightarrow 1$ is optimal. Following the previous analysis we assume that $V_1 = 1$. Also we use $\mathbf{x}_1(t_k) = \mathbf{x}$ for the sake of brevity. We also assume the rewards are linearly decreasing to zero as shown in the blue curve in Figure 2·1; $\phi_i(t) = 1 - \frac{t}{D_i}$. The reward for each of paths can be calculated as below:

$$R_{1\rightarrow 2} = \lambda_1[1 - \frac{d(\mathbf{x}, \mathbf{y}_1)}{D_1}] + \lambda_2[1 - \frac{d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_1 - \mathbf{y}_2)}{D_2}] \tag{2·37}$$

$$R_{2\rightarrow 1} = \lambda_2[1 - \frac{d(\mathbf{x}, \mathbf{y}_2)}{D_2}] + \lambda_1[1 - \frac{d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2 - \mathbf{y}_1)}{D_1}] \tag{2·38}$$

Now let's find the optimality criteria for each path. Assuming $R_{1\rightarrow 2} > R_{2\rightarrow 1}$ we find the criteria for $\lambda_1, \lambda_2, D_1$ and D_2 and the distances between agent and targets.

$$\begin{aligned} &\lambda_1[1 - \frac{d(\mathbf{x}, \mathbf{y}_1)}{D_1}] + \lambda_2[1 - \frac{d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_1, \mathbf{y}_2)}{D_2}] \\ &\quad > \lambda_2[1 - \frac{d(\mathbf{x}, \mathbf{y}_2)}{D_2}] + \lambda_1[1 - \frac{d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2, \mathbf{y}_1)}{D_1}] \end{aligned} \tag{2·39}$$

By rearranging the two side of inequality we have:

$$\begin{aligned} &\lambda_1 + \lambda_2 - \frac{\lambda_1}{D_1} d(\mathbf{x}, \mathbf{y}_1) - \frac{\lambda_2}{D_2}(d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_1, \mathbf{y}_2)) \\ &\quad > \lambda_1 + \lambda_2 - \frac{\lambda_2}{D_2} d(\mathbf{x}, \mathbf{y}_2) - \frac{\lambda_1}{D_1}(d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2, \mathbf{y}_1)) \end{aligned} \tag{2·40}$$

Now we can get simplify the two sides even more:

$$\begin{aligned}\frac{\lambda_1}{D_1}d(\mathbf{x},\mathbf{y}_1)+\frac{\lambda_2}{D_2}[d(\mathbf{x},\mathbf{y}_1)+d(\mathbf{y}_1,\mathbf{y}_2)]\\ <\frac{\lambda_2}{D_2}d(\mathbf{x},\mathbf{y}_2)+\frac{\lambda_1}{D_1}[d(\mathbf{x},\mathbf{y}_2)+d(\mathbf{y}_2,\mathbf{y}_1)]\end{aligned} \tag{2·41}$$

And with more rearranging we have the final inequality:

$$\begin{aligned}\frac{\lambda_1}{D_1}[d(\mathbf{x},\mathbf{y}_1)-d(\mathbf{x},\mathbf{y}_2)+d(\mathbf{y}_2,\mathbf{y}_1)]\\ <\frac{\lambda_2}{D_2}[d(\mathbf{x},\mathbf{y}_2)-d(\mathbf{x},\mathbf{y}_1)+d(\mathbf{y}_1,\mathbf{y}_2)]\end{aligned} \tag{2·42}$$

Theorem 2.2. *In a mission with two target and one agent with the agent located at* $\mathbf{x}_1(t)$*, target 1 and 2 located at* $\mathbf{y}_1$ *and* $\mathbf{y}_2$ *if target i's reward at time t is* $\lambda_i(1-\frac{t}{D_i})$*, the CRH controller correctly finds the optimal path for this mission, assuming* $\gamma=0$ *in* (2·17)*.*

Proof. see the Appendix. □

2.4.6 Monotonicity in the Look Ahead Steps

Questions that come into mind after introducing the multiple look ahead steps algorithm in the CRH controller are: How many look ahead steps should we perform? Is it always better to do more look ahead steps? Or in a simple way, does the three steps look ahead always gives a better answer than the two look ahead steps?

The answer to the first question is that it depends on the size of the problem and our computation capability. We can even adjust the number of look ahead steps during the course of the solution. We can start with more when there is more targets available and lower the number once there is only a few targets in the mission space.

The answer to the other two questions is No. As much as one would like to have a sort of monotonicity effect in this problem, the complexity of the problem and its huge dependence on the topology of the mission causes the non-monotone results with different number of look ahead steps. Here we are going to show a case with 10

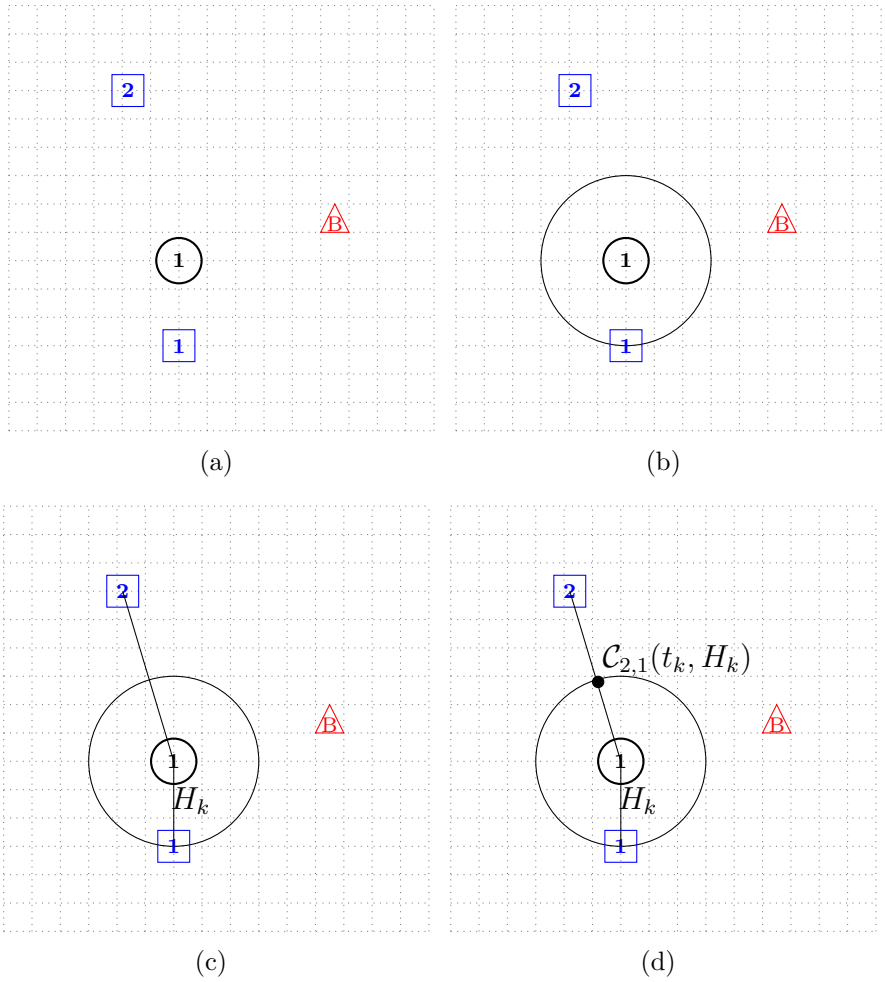
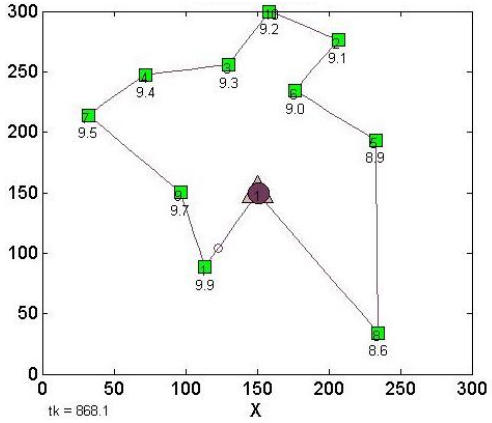


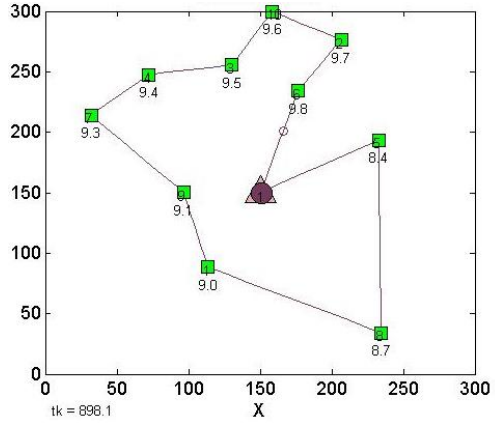
Figure 2·12: Sample mission space with 2 targets and one agent

equally important targets and one agent. This is a straight forward TSP for which the optimal path can be obtained through an exhaustive search. For this case the one and two look ahead steps CRH Controllers find the same path with a reward of 92.6683. However, once we move up to three look ahead steps, the CRH controller degrade to a lower reward path with 92.5253 reward. The path for these controllers is shown in figures 2.13(a) and 2.13(b). The optimal path that is calculated through the exhaustive search is yet to be obtained by the CRH controllers. This happens when we go up to six look ahead steps and the optimal path is calculated correctly (Figure

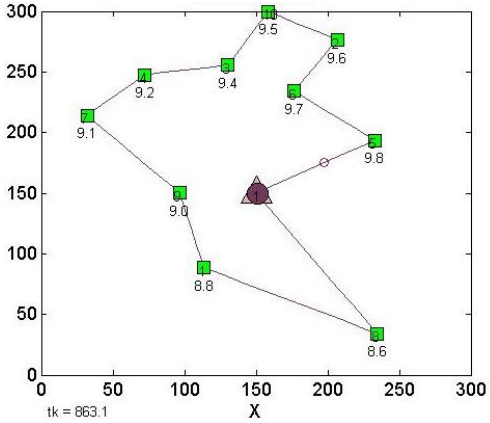
2.13(d)). The observation is that the non-monotone results from higher number of look ahead is a local effect and once we increase the look ahead steps CRH controller it can solve the problem to the optimality which as we see is not the case but hopefully with enough number of look ahead steps we are able to converge to a reward that would be the best we can get with any larger number of look ahead steps.



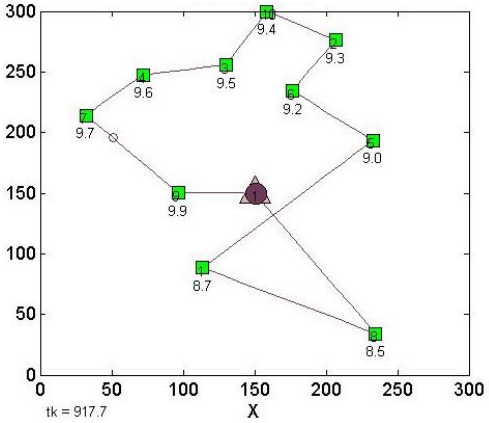
(a) One Step Look Ahead: $[1-9-7-4-3-10-2-6-5-8]$ - Reward=92.6683 - Time=868



(b) Three Step Look Ahead: $[6-2-10-3-4-7-9-1-8-5]$ - Reward=92.5253 - Time=897



(c) Five Step Look Ahead: $[5-6-2-10-3-4-7-9-1-8]$ - Reward=92.6031 - Time=862



(d) Six Step Look Ahead: $[9-7-4-3-10-2-6-5-1-8]$ - Reward=92.7436 - Time=916

Figure 2·13: 10 Target mission with different number of look ahead steps

Table 2.1: TSP benchmark instances comparison with the CRH controller algorithm

TSP Instance	Optimal Tour Length	Two Step Look Ahead	Three Step Look Ahead	Limited Range Agent	Minimum Error (%)
att48	33522	38011	37492	41112	11.8
eil51	426	547	480	507	12.6
berlin52	7542	8713	8713	8137	7.8
st70	675	840	818	816	20.8
eil76	538	633	635	655	17.6
pr76	108159	146980	131678	146944	21.7
rat99	1211	1451	1470	1591	19.8
rd100	7910	9529	9123	9618	15.3
kroA100	21282	25871	24795	23782	11.7
kroB100	22141	28093	27415	28581	23.8
kroC100	20749	24603	25561	26171	18.5

2.5 Simulation Results

We consider several scenarios for single and multiple agent MRCP. The performance of the original and modified CRH are compared. The common simulation parameters $\Delta = 0$, $V_j = 1$, $\alpha_i = 1$ and $\gamma = 0$ are used unless stated otherwise.

2.6 TSP Benchmarks Comparison

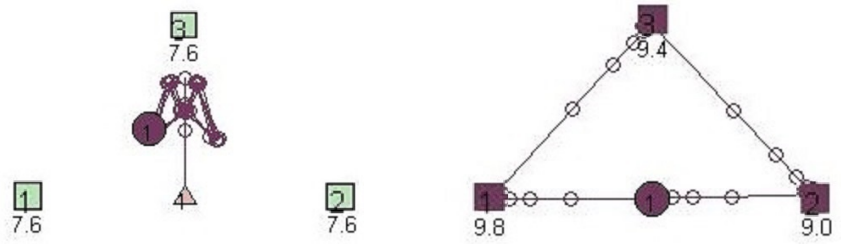
In this section we use the CRH controller as a path planning algorithm for some benchmark TSP problems. The table 2.1 shows the result of the two step and three step look ahead algorithm compared to the optimal results from (Reinelt, 1991). Although the CRH controller algorithm is not necessarily designed for the deterministic TSP problem and will definitely not perform as well as the high efficient TSP algorithms, it still produce a relatively reasonable error for most of the instances.

2.7 Limited Sensing Agents

In an attempt to measure the sensitivity of the results of the modified CRH controller to partial mission information, we tried the same TSP benchmark problems but assuming agents have limited sensing range. In these scenarios, the agent only senses a target if it's within its sensing range. We have assumed the sensing range in each case is equal to 20% of the maximum dimension of the mission space. For instance if a mission space is 300×200 the sensing range of the agent is $\frac{300}{5} = 60$. The result of these missions with partial information is shown in the last column of table 2.1. The best result of two and three look ahead step is presented for this case. The results in most cases are comparable to the full information results. The computation time for the limited range agents is about an order of magnitude shorter than the other one. These results show the low sensitivity of the CRH controller performance to non-local information for each agent. This can be promising for the distributed implementation of the CRH controller and also in cases where targets are not known a priori and should be sensed by the agents locally.

2.8 Addressing Instabilities

In the previous sections we discussed a simple case mission with three linearly discounting targets. The original CRH controller fails to perform this mission and gets stuck in instability. In Figure 2·14 it is shown that the modified CRH can easily find the optimal result for this simple case.



(a) Original CRH Oscillation (b) Modified CRH Optimal Solution

Figure 2·14: Comparison of the two CRH controller on a 3 targets mission

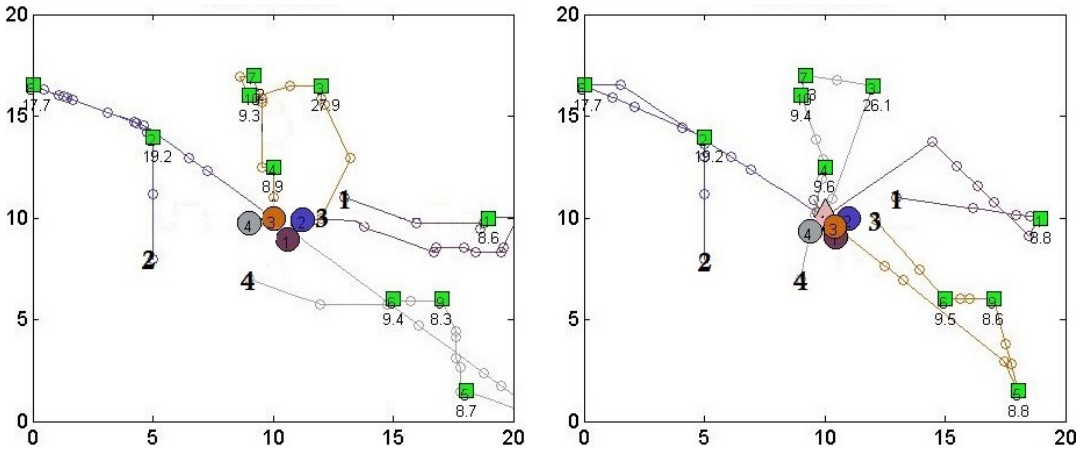
2.9 Comparison with the Previous work

A sample Case from previous work

The simulation example is a previously studied case in (Li and Cassandras, 2006b). In this case there are four vehicles and 10 targets. Targets are shown with squares and the current reward shows below each target. Agents are solid color circles and the base is shown with a triangle in the middle of the mission space. Agents would only go back to the base when all targets are visited. The initial location of each agent is shown with a bold black number corresponding to the agent's ID number (Figures 2.15(a) and 2.15(b)). Each target point has a maximum reward of R_i and a deadline of D_i. For $i = 1, 4, 5, 6, 7, 9, 10$ we have $R_i = 10$ while for $i = 2, 8$ $R_i = 20$ and finally $R_3 = 30$. The deadline for the targets to be visited before the reward goes sharply to zero is for $i = 3, 5, 6, 8$, $D_i = 20$ while for $i = 2, 4, 7, 10$ we have $D_i = 30$. Finally for $i = 1, 9$ $D_i = 10$.

The previous CRH controller in (Li and Cassandras, 2006b) is implemented again to be compared to our new CRH controller. In figures 2.15(a) and 2.15(b) the original and the new CRH controller are implemented on the same mission. For the new CRH a one step look ahead is considered in this case. As it can be observed the original method tend to move the agents not directly toward a target but in the space between

targets. This behavior is specially due to the way the potential function is defined. In both cases the mission tasks for agents 1 and 2 won't change drastically between the two controllers. However, in the original CRH as it can be seen in Figure 2.15(a) agent 1 moves toward the space between targets 1 and 9 in the first move. Then once agent 4 gets closer to the area around target 9, agent 1 decides to collect target 1. For agents 3 and 4 however, the roles almost switch between the two controllers.In the new CRH in the first step the active target set for agent 1 is $\{1, 3, 4, 6\}$, for target 2 is $\{2, 4\}$, for target 3 is $\{1, 4, 6\}$ and for target 4 is $\{4, 6\}$. The decision is then to find the set of four active targets that are optimal in this step. The cases with the same active target for two or more agents won't be considered.



(a) Original CRH Controller , Reward = 126.865 , Mission Time=22.39

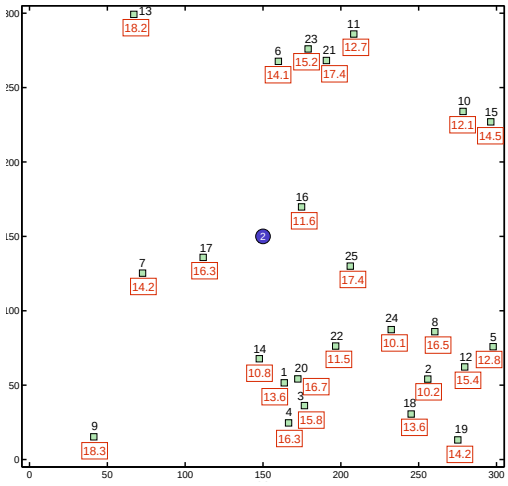
(b) New CRH Controller , Reward = 127.257 , Mission Time=22.42

Figure 2·15: Original and new CRH comparison for a symmetrical 8 target case

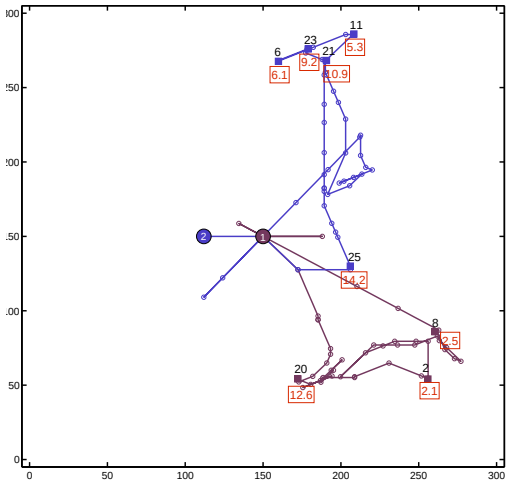
In the final results the new CRH results in a slightly higher reward of 127.257 in 22.42 seconds compared to the original CRH reward of 126.86 in 22.39 seconds. Note that here the time is not limited so the slight higher time in the new CRH doesn't discount the higher reward.

Comparison over a new random case

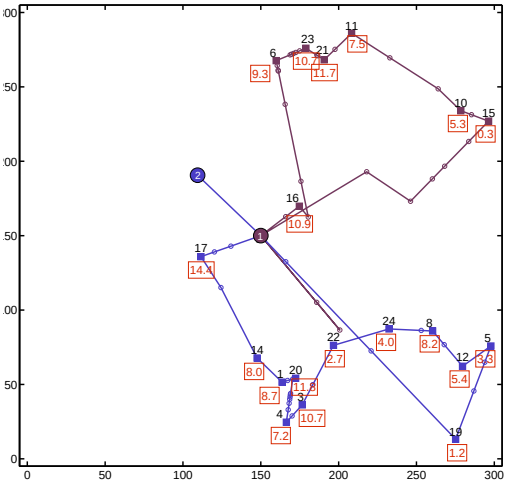
A mission with 25 targets distributed uniformly in the mission space and 2 agents starting at the base is considered. $\lambda_i \sim U(10, 20)$ and $D_i \sim U(300, 600)$ are uniformly distributed. The initial mission is shown in Figure 2.16(a). In this case the the original CRH result shown in Figure 2.16(b)) under-performs comparing to both three and five step look ahead CRH (Figure 2.16(c)) and 2.16(d) by a large margin. The original CRH gets stuck in an oscillatory behavior when agent 1 is close to target 20 and then once around target 8. As we can see in Figure 2.16(d) the mission space is almost divided into a top and bottom partition and the most number of targets are collected by the five step look ahead controller.



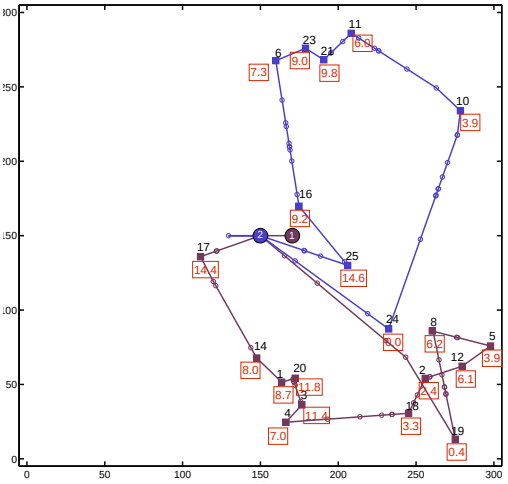
(a) Initial Mission



(b) Original CRH Controller, Reward=62.8, Travel Time=714, Computation Time:108s



(c) 3-L Controller, Reward=141.29, Travel Time=677, Computation Time:104s



(d) 5-L Controller, Reward=143.42, Travel Time=657, Computation Time:1400s

Figure 2·16: Performance comparison of the original and new CRH algorithms (Numbers in red show the reward for each target)

2.9.1 Random Cases Comparison

To compare the overall performance of the modified CRH controller, we generated 10 random missions, each with 20 target that are uniformly located in a 300×300

mission space and two agents initially at the base. $\lambda_i \sim U(2, 12)$ and $D_i = 300$. The results for all runs are shown in table 2.2. We can see that the average total reward is increased by 26% while the average mission time is increased by 8%.

Table 2.2: 20 Target and 2 Agents Random Missions

Mission #	Original CRH		Three Step Look Ahead CRH	
	Total Reward	Mission Time	Total Reward	Mission Time
1	33.92	412	45.24	536
2	41.48	439	30.12	426
3	17.03	476	41.19	483
4	14.08	389	37.24	457
5	21.5	444	47.25	537
6	44.61	389	47.91	471
7	23.93	528	19.61	462
8	16.39	415	24.46	489
9	30.92	478	19.8	429
10	18.8	458	19.4	476
Average	26.26	443	33.22	479

2.10 Sparsity Factor in Clustered Missions

We considered 8 random mission with 20 targets that are once located uniformly and in another case located in 9 clusters. We want to see if the sparsity factory ζ_i in (2·17) would help us in either of these cases. The mission space is 300×300 with rewards and deadlines the same as the previous case. We once solve the missions with a $\gamma = 0$ which eliminates the effect of ζ_i and then run them with $\gamma = 0.3$. The results show that in the clustered missions rewards are improved by about 24% margin whereas in the uniform cases the reward in both cases is equal on average.

Table 2.3: Effect of the sparsity factor for clustered missions ζ_i

Mission #	$\gamma = 0$		$\gamma = 0.3$	
	Total Reward	Mission Time	Total Reward	Mission Time
1	40.62	552	61.9	413
2	64.89	447	64.64	420
3	35.24	471	63.8	461
4	63.78	465	64.64	478
5	25.42	493	26.5	449
6	22	454	22	454
7	44.1	458	46.84	449
8	34.26	466	61.21	472
Average	41.29	475	51.44	449

Chapter 3

Event Excitation in Multi-agent Systems

3.1 General Framework for Multi-agent Systems

Multi-agent systems are commonly modeled as hybrid systems with time-driven dynamics describing the motion of the agents or the evolution of physical processes in a given environment, while event-driven behavior characterizes events that may occur randomly (e.g., an agent failure) or in accordance to control policies (e.g., an agent stopping to sense the environment or to change directions). In some cases, the solution of a multi-agent dynamic optimization problem is reduced to a policy that is naturally parametric. As such, a multi-agent system can be studied with parameterized controllers aiming to meet certain specifications or to optimize a given performance metric. Moreover, in cases where such a dynamic optimization problem cannot be shown to be reduced to a parametric policy, using such a policy is still near-optimal or at least offers an alternative.

In order to build a general framework for multi-agent optimization problems, assuming S as the mission space, we introduce the function $R(w) : S \to \mathbb{R}$ as a "property" of point $w \in S$. For instance, $R(w)$ could be a weight that gives relative importance to one point in S compared to another. Setting $R(w) > 0$ for only a finite number of points implies that we limit ourselves to a finite set of points of interest while the rest of S has no significant value.

Assuming F to be the set of all feasible agent states, We define $P(w, s) : S \times F \to \mathbb{R}$ to capture the cost/reward resulting from how agents with state $s \in F$ interact

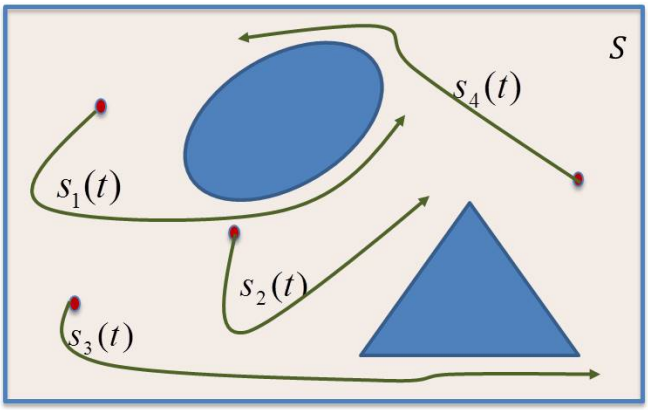


Figure 3·1: Multi-agent system in a dynamic setting, blue areas are obstacles

with $w \in S$. For instance, in coverage problems if an "event" occurs at w, then $P(w, s)$ is the probability of agents jointly detecting such events based on the relative distance of each agent from w.

In general settings, the objective is to find the best state vector $s_1, \cdots, s_N$ so that N agents achieve a maximal reward (minimal cost) from interacting with the mission space S:

$$\min_{s \in F} J = \int_S P(w, s) R(w) dw \tag{3·1}$$

This static problem can be extended to a dynamic version where the agents determine optimal trajectories $s_i(t)$, $t \in [0, T]$, rather than static states:

$$\min_{u(t) \in U} J = \int_0^T \int_S P(w, \mathbf{s}(u(t))) R(w, t) dw dt \tag{3·2}$$

subject to motion dynamics:

$$\dot{s}_j(t) = f_j(s_j, u_j, t), \ j = 1, \cdots, N \tag{3·3}$$

In Figure 3·1, such a dynamic multi agent system is illustrated.

As an example, consensus problems are just a special case of (3·1). Suppose that

we consider a finite set of points $w \in S$ which coincide with the agents states $s_1, ..., s_N$ (which are not necessarily their locations). Then we can set $P(w, s) = \|s_i - s_j\|^2$ and, therefore, replace the integral in (3·1) by a sum. In this case, $R(w) = R_i$ is just the weight that an agent carries in the consensus algorithm. An optimum occurs when $\|s_i - s_j\|^2 = 0$ for all i, j, i.e., all agents "agree" and consensus is reached. This is a special case because of the simplicity in $P(w, s)$ making the problem convex so that a global optimum can be achieved, in contrast to most problems we are interested in.

As for the formulation in (3·2), consider a trajectory planning problem where N mobile agents are tasked to visit M stationary targets in the mission space S. Target behavior is described through state variables $x_i(t)$ which may model reward functions, the amount of data present at i, or other problem-dependent target properties. More formally, let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space and $\omega \in \Omega$ a realization of the system where target dynamics are subject to random effects:

$$\dot{x}_i(t) = g_i(x_i(t), \omega) \tag{3·4}$$

$g_i(\cdot)$ is as such that $x_i(t)$ is monotonically increasing by t and it resets to zero each time a target is completely emptied by an agent. In the context of (3·2), we assume the M targets are located at points $w_i,\ i = 1, \cdots, M$ and define

$$R(w, t) = \begin{cases} R(x_i(t), w) & \text{if } w \in C(w_i) \\ 0 & \text{otherwise} \end{cases} \tag{3·5}$$

to be the value of point w, where $C(w_i)$ is a compact 2-manifold in $\mathbb{R}^2$ containing w_i which can be considered to be a region defined by the sensing range of that target relative to agents (e.g., a disk centered at w_i). Note that $R(w, t)$ is also a random variable defined on the same probability space above. Given that only points $w \in C(w_i)$ have value for the agents, there is an infinite number of points $w \notin C(w_i)$ such that $R(w, t) = 0$ provided the following condition holds:

Condition 1: If $\exists i$ such that $w \in C(w_i)$ then $w \notin C(w_j)$ holds $\forall j \neq i$.

This condition is to ensure that two targets do not share any point w in their respective sensing ranges. Also it ensures that the set $\{C(w_i) \mid i = 1 : \cdots, M\}$ does not create a compact partitioning of the mission space and there exist points w which do not belong to any of the $C(w_i)$.

Viewed as a stochastic hybrid system, we may define different modes depending on the states of agents or targets and events that cause transitions between these modes. Relative to a target i, any agent has at least two modes: being at a point $w \in C(w_i)$, i.e., visiting this target or not visiting it. Within each mode, agent j's dynamics, dictated by (3·3), and target i's dynamics in (3·4) may vary. Accordingly, there are at least two types of events in such a system: (i) δ^0_{ij} events occur when agent j initiates a visit at target i, and (ii) δ^+_{ij} events occur when agent j ends a visit at target i. Additional event types may be included depending on the specifics of a problem, e.g., mode switches in the target dynamics or agents encountering obstacles.

An example is shown in Figure 3·2, where target sensing ranges are shown with green circles and agent trajectories are shown in dashed lines starting at a base shown by a red triangle. In the blue trajectory, agent 1 moves along the trajectory that passes through points $A \rightarrow B \rightarrow C \rightarrow D$. It is easy to see that when passing through points A and C we have δ^0_{i1} and $\delta^0_{i'1}$ events, while passing through B and D we have δ^+_{i1} and $\delta^+_{i'1}$ events. The red trajectory is an example where none of the events is excited. Suppose we consider an on-line trajectory adjustment process in which the agent improves its trajectory based on its performance measured through (3·5). In this case, $R(w,t) = 0$ over all t, as long as the agent keeps using the red trajectory, i.e., no event ever occurs. Therefore, if an event-driven approach is used to control the trajectory adjustment process, no action is ever triggered and the approach is ineffective. In contrast, in the blue trajectory the controller can extract

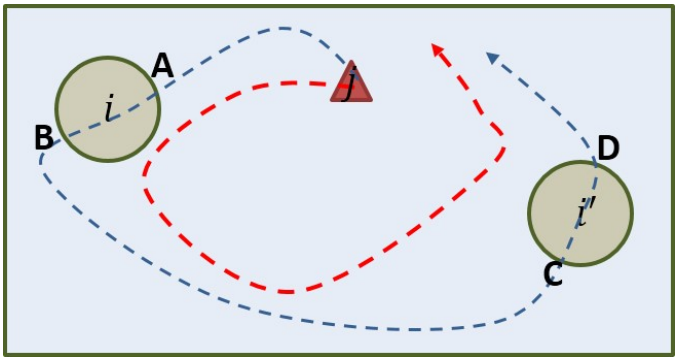


Figure 3·2: Sample trajectories

useful information from every observed event; such information (e.g., a gradient of J with respect to controllable parameters as described in the next section) can be used to adjust the current trajectory so as to improve the objective function J in (3·1) or (3·2).

Therefore, if we are to build an optimization framework for this class of stochastic hybrid systems to allow the application of event-driven methods by calculating a performance measure gradient, then a fundamental property required is the occurrence of at least some events in a sample realization. In particular, the Infinitesimal Perturbation Analysis (IPA) method uses a single sample realization of the system over which events are observed along with their occurrence times and associated system states. Suppose that the trajectories can be controlled through a set of parameters forming a vector Θ. Then, under some mild assumptions, IPA provides an unbiased estimate of the gradient of a performance metric $J(\Theta)$ with respect to Θ. This gradient is then used to improve the trajectory and ultimately seek an optimal one when appropriate conditions hold. As in the example of Figure 3·2, it is possible to encounter trajectory realizations where no events occur in the system. In the above example, this can easily happen if the trajectory does not pass through any target. The existence of such undesirable trajectories is the direct consequence of Condition

1. This lack of event excitation results in event-based controllers being unsuitable.

New Metric: In order to overcome this issue we propose a new definition for $R(w,t)$ in (3·5) as follows:

$$R(w,t) = \sum_{i=1}^{M} h_i(x_i(t), d_i(w)) \tag{3·6}$$

where $w \in S$, $h_i(\cdot)$ is a function of the target's state $x_i(t)$ and $d_i(w) = \|w_i - w\|$. Note that, if $h_i(\cdot)$ is properly defined, (3·6) yields $R(w,t) > 0$ at all points.

While the exact form of $h_i(\cdot)$ depends on the problem, we impose the condition that $h_i(\cdot)$ is monotonically decreasing in $d_i(w)$. We can think of $h_i(\cdot)$ as a value function associated with point w_i. Using the definition of $R(w,t)$, this value is spread out over all points $w \in S$ rather than being concentrated at the single point w_i. This creates a continuous potential field for the agents leading to a non-zero gradient of the performance measure even when the trajectories do not excite any events. This non-zero gradient will then induce trajectory adjustments that naturally bring them toward ones with observable events.

Finally, recalling the definition in (3·2), we also define:

$$P(w,s) = \sum_{j=1}^{N} \|s_j(t) - w\|^2 \tag{3·7}$$

the total quadratic travel cost for agents to visit point w.

In Section 3.3, we will show how to apply $R(w,t)$ and $P(w,s)$ defined as above in order to determine optimal agent trajectories for a class of multi-agent problems of the form (3·2). First, however, we review in the next section the event-driven IPA calculus which allows us to estimate performance gradients with respect to controllable parameters.

3.2 Event-driven IPA Calculus

In this section, we go through a brief introduction of the infinitesimal perturbation method (IPA). IPA is an event-driven method that allows us to calculate an unbiased estimate for the performance metric of a stochastic hybrid system (SHS) with respect to the control variables of the system. In general, we define $\theta \in F_\theta$ to be the control parameter of the system where F_θ is a compact convex set. We fix a particular value of the parameter $\theta \in F_\theta$ and assuming all the random processes are defined on a common probability space, we study a resulting sample path of this SHS. In this sample realization, let $\tau_k(\theta)$, $k = 1, 2, \cdots$ denote the occurrence times of the discrete events in increasing order, and for convenience define $\tau_0(\theta) = 0$.

We will omit the dependency on θ when no confusion arises. The continuous state of the SHS is generally a function of θ, and time t, and is thus denoted by $x(\theta, t)$. The SHS is at a discrete mode for each interval $[\tau_k(\theta), \tau_{k+1}(\theta))$, and the time-driven state satisfies

$$\dot{x} = f_k(x, \theta, t) \tag{3·8}$$

in which x is any of the continuous state variables of the system and $\dot{x}$ denotes $\frac{\partial x}{\partial t}$

Perturbation analysis studies the sensitivity of the state $x(\theta, t)$ and the event times $\tau_k(\theta)$ with respect to θ and, ultimately, how θ influences performance metrics of the problem.

The discrete mode of the system will possibly change at each event τ_k which then results in changes in some of the continuous dynamics of the system. The event times τ_k play an important role in defining the interactions between the time-driven and event-driven dynamics of the system.

Following the framework in (Cassandras et al., 2010), consider a general perfor-

mance function J of the control parameter θ:

$$J(\theta; x(\theta,0), T) = E[L(\theta); x(\theta,0), T)] \tag{3·9}$$

where $L(\theta; x(\theta,0), T)$ is a sample function of interest evaluated in the interval $[0,T]$ with initial conditions $x(\theta,0)$. For simplicity, we write $J(\theta)$ and $L(\theta)$. Suppose that there are K events, with occurrence times generally dependent on θ, during the time interval $[0,T]$ and define $\tau_0 = 0$ and $\tau_{K+1} = T$. We assume $L_k : \mathbb{R}^n \times F_\theta \times \mathbb{R}^+ \to \mathbb{R}$ be a function that describes the realization in the interval $[\tau_k, \tau_{k+1}]$. Now we can define $L(\theta)$ by

$$L(\theta) = \sum_{k=0}^{K} \int_{\tau_k}^{\tau_{k+1}} L_k(x, \theta, t)dt \tag{3·10}$$

The restriction of the definition of $J(\theta)$ to a finite horizon T which is independent of θ is only for the sake of simplicity and can be changed based on different problems.

Returning to the stochastic setting, the ultimate goal of the iterative process shown is to optimize $E_\omega[L(\theta,\omega)]$, where we use ω to emphasize dependence on a sample path ω of a SHS (clearly, this is reduced to $L(\theta)$ in the deterministic case). Achieving such optimality is possible under standard ergodicity conditions imposed on the underlying stochastic processes, as well as the assumption that a single global optimum exists; otherwise, the gradient-based approach is simply continuously attempting to improve the observed performance $L(\theta,\omega)$. Thus, we are interested in estimating the gradient

$$\frac{dJ(\theta)}{d\theta} = \frac{dE_\omega[L(\theta,\omega)]}{d\theta} \tag{3·11}$$

by evaluating $\frac{dL(\theta,\omega)}{d\theta}$ based on directly observed data. We obtain θ^* by optimizing $J(\theta)$ through an iterative scheme of the form

$$\theta_{n+1} = \theta_n - \eta_n H_n(\theta_n; x(\theta,0), T, \omega_n),\ n = 0, 1, \cdots \tag{3·12}$$

where η_n is a step size sequence and $H_n(\theta_n; x(\theta,0), T, \omega_n)$ is the estimate of $\frac{dJ(\theta)}{d\theta}$ at $\theta = \theta_n$. In using IPA, $H_n(\theta_n; x(\theta,0), T, \omega_n)$ is the sample derivative $\frac{dL(\theta,\omega)}{d\theta}$, which is an unbiased estimate of $\frac{dJ(\theta)}{d\theta}$ if the condition (dropping the symbol ω for simplicity)

$$E[\frac{dL(\theta)}{d\theta}] = \frac{dE[L(\theta)]}{d\theta} = \frac{dJ(\theta)}{d\theta} \tag{3·13}$$

is satisfied, which turns out to be the case under mild technical conditions. The conditions under which algorithms of the form (3·12) converge are well-known (e.g., see (Kushner and Yin, 2003)). In addition to the unbiasedness property, it can be shown that such gradient estimates are independent of the stochastic processes of the underlying SHS and require minimal information from the observed sample path.

The process through which IPA evaluates $\frac{dL(\theta)}{d\theta}$ is based on analyzing how changes in θ influence the state $x(\theta,t)$ and the event times $\tau_k(\theta)$. In turn, this provides information on how $L(\theta)$ is affected, because it is generally expressed in terms of these variables. Given $\theta = [\theta_1, ..., \theta_l]^T$, we use the Jacobian matrix notation:

$$x'(\theta,t) = \frac{\partial x(\theta,t)}{\partial \theta},\ {\tau_k}' = \frac{\partial \tau_k(\theta)}{\partial \theta}, k = 1, \cdots, K \tag{3·14}$$

for all state and event time derivatives. For simplicity of notation, we omit θ from the arguments of the functions above unless it is essential to stress this dependence. It is shown in (Cassandras et al., 2010) that $x'(t)$ satisfies:

$$\frac{dx'(t)}{dt} = \frac{\partial f_k(t)}{\partial x} x'(t) + \frac{\partial f_k(t)}{\partial \theta} \tag{3·15}$$

for $t \in [\tau_k(\theta), \tau_{k+1}(\theta))$ with boundary condition

$$x'(\tau_k^+) = x'(\tau_k^-) + [f_{k-1}(\tau_k^-) - f_k(\tau_k^+)]\tau_k' \tag{3·16}$$

for $k = 0, \cdots, K$. We note that whereas $x(t)$ is often continuous in t, $x'(t)$ may be

discontinuous in t at the event times τ_k; hence, the left and right limits above are generally different. If $x(t)$ is not continuous in t at $t = \tau_k(\theta)$, the value of $x(\tau_k^+)$ is determined by the state reset function $r(q, q', x, \nu, \delta)$ and

$$x'(\tau_k^+) = \frac{dr(q, q', x, \nu, \delta)}{d\theta} \tag{3·17}$$

Furthermore, once the initial condition $x'(\tau_k^+)$ is given, the linearized state trajectory $x'(t)$ can be computed in the interval $t \in [\tau_k(\theta), \tau_{k+1}(\theta))$ by solving (3·15) to obtain

$$x'(t) = e^{\int_{\tau_k}^t \frac{\partial f_k(u)}{\partial x} du} \Big[\int_{\tau_k}^t \frac{\partial f_k(v)}{\partial \theta} e^{-\int_{\tau_k}^t \frac{\partial f_k(u)}{\partial x} du} dv + \xi_k \Big] \tag{3·18}$$

with the constant ξ_k determined from $x'(\tau_k^+)$. In order to complete the evaluation of $x'(\tau_k^+)$ we need to also determine τ_k'. If the event at $\tau_k(\theta)$ is exogenous $\tau_k' = 0$ and if the event at $\tau_k(\theta)$ is endogenous:

$$\tau_k' = -\Big[\frac{\partial g_k}{\partial x} f_k(\tau_k^-)\Big] \Big(\frac{\partial g_k}{\partial \theta} + \frac{\partial g_k}{\partial x} x'(\tau_k^-)\Big) \tag{3·19}$$

where $g_k(x, \theta) = 0$ and it is defined as long as $\frac{\partial g_k}{\partial x} f_k(\tau_k^+) \neq 0$ (details may be found in (Cassandras et al., 2010).)

The derivative evaluation process involves using the IPA calculus in order to evaluate the IPA derivative $\frac{dL}{d\theta}$. This is accomplished by taking derivatives in (3·10) with respect to θ:

$$\frac{dL(\theta)}{d\theta} = \sum_{k=0}^{K} \frac{d}{d\theta} \int_{\tau_k}^{\tau_{k+1}} L_k(x, \theta, t) dt \tag{3·20}$$

Applying the Leibnitz rule, we obtain, for every $k = 0, \cdots, K$,

$$\begin{aligned} &\frac{d}{d\theta} \int_{\tau_k}^{\tau_{k+1}} L_k(x, \theta, t) dt \\ &\quad = \int_{\tau_k}^{\tau_{k+1}} \Big[\frac{\partial L_k(x, \theta, t)}{\partial x} x'(t) + \frac{\partial L_k(x, \theta, t)}{\partial \theta}\Big] dt \\ &\quad + L_k(x(\tau_{k+1}), \theta, \tau_{k+1}) \tau_{k+1}' - L_k(x(\tau_k), \theta, \tau_k) \tau_k' \end{aligned} \tag{3·21}$$

In summary the three equations (3·16), (3·18) and (3·19) form the basis of the IPA calculus and allow us to calculate the final derivative in (3·21). In the next section IPA is applied to a data collection problem in a multi-agent system.

3.3 The Data Collection Problem

We consider a class of multi-agent problems where the agents must cooperatively visit a set of target points to collect associated rewards (e.g., to collect data that are buffered at these points.). The mission space is $S \subset \mathbb{R}^2$. This class of problems falls within the general formulation introduced in (3·2). The state of the system is the position of agent j time t, $s_j(t) = [s_j^x(t), s_j^y(t)]$ and the state of the target i, $x_i(t)$. The agent's dynamics (3·3) follow a single integrator:

$$\dot{s}_j^x(t) = u_j(t)\cos\theta_j(t), \qquad \dot{s}_j^y(t) = u_j(t)\sin\theta_j(t) \tag{3·22}$$

where $u_j(t)$ is the scalar speed of the agent (normalized so that $0 \leq u_j(t) \leq 1$) and $\theta_j(t)$ is the angle relative to the positive direction, $0 \leq \theta_j(t) < 2\pi$. Thus, we assume that each agent controls its speed and heading.

We assume the state of the target $x_i(t)$ represents the amount of data that is currently available at target i (this can be modified to different state interpretations). The dynamics of $x_i(t)$ in (3·4) for this problem are:

$$\dot{x}_i(t) = \begin{cases} 0 & \text{if } x_i(t) = 0 \text{ and } \sigma_i(t) \leq \mu_{ij}p(s_j(t), w_i) \\ \sigma_i(t) - \mu_{ij}p(s_j(t), w_i) & \text{otherwise} \end{cases} \tag{3·23}$$

i.e., we model the data at the target as satisfying simple flow dynamics with an exogenous (generally stochastic) inflow $\sigma_i(t)$ and a controllable rate with which an agent empties the data queue given by $\mu_{ij}p(s_j(t), w_i)$. For brevity we set $p(s_j(t), w_i) = p_{ij}(t)$ which is the normalized data collection rate from target i by agent j and μ_{ij} is a nominal rate corresponding to target i and agent j.

Assuming M targets are located at $w_i \in S$, $i = 1, \ldots, M$, and have a finite range of r_i, then agent j can collect data from w_i only if $d_{ij}(t) = \|w_i - s_j(t)\| \leq r_i$. We then assume that: (**A1**) $p_{ij}(t) \in [0, 1]$ is monotonically non-increasing in the value of $d_{ij}(t) = \|w_i - s_j(t)\|$, and (**A2**) it satisfies $p_{ij}(t) = 0$ if $d_{ij}(t) > r_i$. Thus, $p_{ij}(t)$ can model communication power constraints which depend on the distance between a data source and an agent equipped with a receiver (similar to the model used in (Ny et al., 2008)) or sensing range constraints if an agent collects data using on-board sensors. For simplicity, we will also assume that: (**A3**) $p_{ij}(t)$ is continuous in $d_{ij}(t)$ and (**A4**) only one agent at a time is connected to a target i even if there are other agents l with $p_{il}(t) > 0$; this is not the only possible model, but we adopt it based on the premise that simultaneous downloading of packets from a common source creates problems of proper data reconstruction. This means that j in (3·23) is the index of the agent that is connected to target i at time t.

The dynamics of $x_i(t)$ in (3·23) results in two new event types added to what was defined earlier, (i) ξ_i^0 events occur when $x_i(t)$ reaches zero, and (ii) ξ_i^+ events occur when $x_i(t)$ leaves zero.

The performance measure is the total content of data left at targets at the end of a finite mission time T. Thus, we define $J_1(t)$ to be the following (recalling that $\{\sigma_i(t)\}$ are random processes):

$$J_1(t) = E\Big[\sum_{i=1}^{M} \alpha_i x_i(t)\Big] \tag{3·24}$$

where α_i is a weight factor for target i. We can now formulate a stochastic optimization problem **P3.1** where the control variables are the agent speeds and headings denoted by the vectors $\mathbf{u}(t) = [u_1(t), \ldots, u_N(t)]$ and $\theta(t) = [\theta_1(t), \ldots, \theta_N(t)]$ respec-

tively (omitting their dependence on the full system state at t).

$$\mathbf{P3.1}: \qquad \min_{\mathbf{u}(t),\theta(t)} J(T) = \frac{1}{T}\int_0^T J_1(t)dt \tag{3·25}$$

where $0 \leq u_j(t) \leq 1$, $0 \leq \theta_j(t) < 2\pi$, and T is a given finite mission time. This problem can be readily placed into the general framework (3·2). In particular, the right hand side of (3·25) is:

$$\begin{aligned} &\frac{1}{T}E\left[\int_0^T \sum_i \int_{C(w_i)} \frac{\alpha_i}{\pi r_i^2} x_i(t)dwdt\right] \\ =&\frac{1}{T}E\left[\int_0^T \int_S \sum_i \frac{\alpha_i \mathbb{1}\{w \in C(w_i)\}}{\pi r_i^2} x_i(t)dwdt\right] \end{aligned} \tag{3·26}$$

This is now in the form of the general framework in (3·2) with

$$R(w,t) = \sum_i \frac{\alpha_i \mathbb{1}\{w \in C(w_i)\}}{\pi r_i^2} x_i(t) \tag{3·27}$$

and

$$P(s_j(t), w) = 1 \tag{3·28}$$

Recalling the definition in (3·5), only points within the sensing range of each target have non-zero values, while all other point value are zero, which is the case in (3·27) above. In addition, (3·28) simply shows that there is no meaningful dynamic interaction between an agent and the environment.

Problem **P3.1** is a finite time optimal control problem. In order to solve this, following previous work in (Khazaeni and Cassandras, 2015) we proceed with a standard Hamiltonian analysis leading to a Two Point Boundary Value Problem (TPBVP) (Bryson and Ho, 1975). Here we present the Hamiltonian analysis for the problem in (3·25). The states and costates are known at $t = 0$ and $t = T$ respectively. We define

a state vector and the associated costate vector:

$$\mathbf{X}(t) = [x_1(t), \ldots, x_M(t), s_1^x(t), s_1^y(t), \ldots, s_N^x(t), s_N^y(t)] \tag{3·29}$$

$$\lambda(t) = [\lambda_1(t), \ldots, \lambda_M(t), \eta_1^x(t), \eta_1^y(t), \ldots, \eta_N^x(t), \eta_N^y(t)] \tag{3·30}$$

Then the Hamiltonian is

$$\begin{aligned} H(\mathbf{X}, \lambda, \mathbf{u}, \theta) &= \frac{1}{T} J_1(t) + \sum_i \lambda_i(t) \dot{x}_i(t) \\ &+ \sum_j \left(\eta_j^x(t) u_j(t) \cos\theta_j(t) + \eta_j^y(t) u_j(t) \sin\theta_j(t) \right) \end{aligned} \tag{3·31}$$

where the costate equations are

$$\dot{\lambda}_i(t) = -\frac{\partial H}{\partial x_i} = -\frac{\alpha_i}{T}, \ \lambda_i(T) = 0 \tag{3·32}$$

$$\dot{\eta}_j^x(t) = -\frac{\partial H}{\partial s_j^x} = -\sum_i \lambda_i(t) \frac{\partial}{\partial s_j^x} \dot{x}_i(t) \tag{3·33}$$

$$\dot{\eta}_j^y(t) = -\frac{\partial H}{\partial s_j^y} = -\sum_i \lambda_i(t) \frac{\partial}{\partial s_j^y} \dot{x}_i(t) \tag{3·34}$$

$$\eta_j^x(T) = \eta_j^y(T) = 0 \tag{3·35}$$

From (3·31), after some trigonometric manipulations, we get

$$\begin{aligned} H(\mathbf{X}, \lambda, \mathbf{u}, \theta) &= \frac{J_1(t)}{T} + \sum_i \lambda_i(t) \dot{x}_i(t) \\ &+ \sum_j u_j(t) \mathrm{sgn}(\eta_j^y(t)) \sqrt{\eta_j^x(t)^2 + \eta_j^y(t)^2} \sin(\theta_j(t) + \psi_j(t)) \end{aligned} \tag{3·36}$$

where $\tan\psi_j(t) = \frac{\eta_j^x(t)}{\eta_j^y(t)}$ for $\eta_j^y(t) \neq 0$ and $\psi_j(t) = \mathrm{sgn}(\eta_j^x(t)) \frac{\pi}{2}$ if $\eta_j^y(t) = 0$. Applying the Pontryagin principle to (3·31) with $(\mathbf{u}^*, \theta^*)$ being the optimal control, we have:

$$H(\mathbf{X}^*, \lambda^*, \mathbf{u}^*, \theta^*) = \min_{\mathbf{u}(t), \theta(t)} H(\mathbf{X}, \lambda, \mathbf{u}, \theta) \tag{3·37}$$

From (3·36) we easily see that we can always make the $u_j(t)$ multiplier to be negative, hence, recalling that $0 \leq u_j(t) \leq 1$,

$$u_j^*(t) = 1 \tag{3·38}$$

Following the Hamiltonian definition in (3·31) we have:

$$\frac{\partial H}{\partial \theta_j} = -\eta_j^x(t) u_j(t) \sin\theta_j(t) + \eta_j^y(t) u_j(t) \cos\theta_j(t) \tag{3·39}$$

and setting $\frac{\partial H}{\partial \theta_j} = 0$ the optimal heading $\theta_j^*(t)$ should satisfy:

$$\tan\theta_j^*(t) = \frac{\eta_j^y(t)}{\eta_j^x(t)} \tag{3·40}$$

Hence, we only need to evaluate $\theta_j^*(t)$ for all $t \in [0, T]$. This can be accomplished by discretizing the problem in time and numerically solving a TPBVP with a forward integration of the state and a backward integration of the costate.

This TPBVP is computationally expensive and easily becomes intractable when problem size grows. The ultimate solution of the TPBVP is a set of agent trajectories that can be put in a parametric form defined by a parameter vector Θ and then optimized over Θ. If the parametric trajectory family is broad enough, we can recover the true optimal trajectories; otherwise, we can approximate them within some acceptable accuracy. Moreover, adopting a parametric family of trajectories and seeking an optimal one within it has additional benefits: it allows trajectories to be periodic, often a desirable property, and it allows one to restrict solutions to trajectories with desired features that the true optimal may not have, e.g., smoothness properties to achieve physically feasible agent motion.

Parameterizing the trajectories and using gradient based optimization methods, in light of the discussions from the previous sections, enables us to make use of Infinitesimal Perturbation Analysis (IPA) to carry out the trajectory optimization

process. We represent each agent's trajectory through general parametric equations

$$s_j^x(t) = f_x(\Theta_j, \rho_j(t)), \qquad s_j^y(t) = f_y(\Theta_j, \rho_j(t)) \tag{3·41}$$

where the function $\rho_j(t)$ controls the position of the agent on its trajectory at time t and Θ_j is a vector of parameters controlling the shape and location of the trajectory. Let $\Theta = [\Theta_1, \ldots, \Theta_N]$. We now revisit problem **P3.1** in (3·25):

$$\min_{\Theta \in F_\Theta} J(\Theta, T) = \frac{1}{T} \int_0^T J_1(\Theta, t) dt \tag{3·42}$$

and will bring in the equations that were introduced in the previous section in order to calculate an estimate of $\frac{dJ(\Theta)}{d\Theta}$ as in (3·11). For this problem due to the continuity of $x_i(t)$ the last two terms in (3·21) vanish. From (3·24) we have:

$$\frac{d}{d\Theta} \int_{\tau_k}^{\tau_{k+1}} \sum_{i=1}^{M} \alpha_i x_i(\Theta, t) dt = \int_{\tau_k}^{\tau_{k+1}} \sum_{i=1}^{M} \alpha_i x_i'(\Theta, t) dt \tag{3·43}$$

In summary, the evaluation of (3·43) requires the state derivatives $x_i'(t)$ explicitly and $s_j'(t)$ implicitly, (dropping the dependence on Θ for brevity). The latter are easily obtained for any specific choice of f and g in (3·41). The former require a rather laborious use of (3·16),(3·18),(3·19) which, reduces to a simple set of state derivative dynamics as shown next.

Proposition 3.1. *After an event occurrence at $t = \tau_k$, the state derivatives $x_i'(\tau_k^+)$ with respect to the controllable parameter Θ satisfy the following:*

$$x_i'(\tau_k^+) = \begin{cases} 0 & \text{if } e(\tau_k) = \xi_i^0 \\ x_i'(\tau_k^-) - \mu_{il}(t) p_{il}(\tau_k) \tau_k' & \text{if } e(\tau_k) = \delta_{ij}^+ \\ x_i'(\tau_k^-) & \text{otherwise} \end{cases}$$

where $l \neq j$ with $p_{il}(\tau_k) > 0$ if such l exists and $\tau_k' = \frac{\partial d_{ij}(s_j)}{\partial s_j} s_j' \left(\frac{\partial d_{ij}(s_j)}{\partial s_j} \dot{s}_j(\tau_k) \right)^{-1}$.

Proof. The proof for this will be presented for a general case of the problem in proposition 4.1. □

As is obvious from Proposition 1, the evaluation of $x_i'(t)$ is entirely dependent on the occurrence of events ξ_i^0 and δ_{ij}^+ in a sample realization, i.e., ξ_i^0 and δ_{ij}^+ cause jumps in this derivative which carry useful information. Otherwise, $x_i'(\tau_k^+) = x_i'(\tau_k^-)$ is in effect and these gradients remain unchanged. However, we can easily have realizations where no events occur in the system (specifically, events of type δ_{ij}^0 and δ_{ij}^+) if the trajectory of agents in the sample realization does not pass through any target. This lack of event excitation results in the algorithm in (3·12) to stall.

In the next section we overcome the problem of no event excitation using the definitions in (3·6) and (3·7). We accomplish this by adding a new metric to the objective function that generates a non-zero sensitivity with respect to Θ.

3.3.1 Event Excitation

Our goal here is to select a function $h_i(\cdot)$ in (3·6) with the property of "spreading" the value of $x_i(t)$ over all $w \in S$. We begin by determining the convex hull produced by the targets, since the trajectories need not go outside this convex hull. Let $\mathcal{T} = \{w_1, w_2, \cdots, w_M\}$ be the set of all target points. Then, the convex hull of these points is as the following:

$$\mathcal{C} = \left\{ \sum_{i=1}^{M} \beta_i w_i | \sum_i \beta_i = 1, \forall i,\ \beta_i \geq 0 \right\} \tag{3·44}$$

Given that $\mathcal{C} \subset S$, we seek some $R(w,t)$ that satisfies the following property for constants $c_i > 0$:

$$\int_{\mathcal{C}} R(w,t)dw = \sum_{i=1}^{M} c_i x_i(t) \tag{3·45}$$

so that $R(w,t)$ can be viewed as a continuous density defined for all points $w \in \mathcal{C}$ which results in a total value equivalent to a weighted sum of the target states $x_i(t)$, $i = 1, \ldots, M$. In order to select an appropriate $h(x_i(t), d_i(w))$ in (3·6), we first define

$d_i^+(w) = \max(\|w - w_i\|, r_i)$ where r_i is the target's sensing range. We then define:

$$R(w,t) = \sum_{i=1}^{M} \frac{\alpha_i x_i(t)}{d_i^+(w)} \tag{3·46}$$

Here, we are spreading a target's reward (numerator) over all w so as to obtain the "total weighted reward density" at w. Note that $d_i^+(w) = \max(\|w - w_i\|, r_i) > 0$ to ensure that the target reward remains positive and fixed for points $w \in C(w_i)$. In order to illustrate this term Figure 3.4(a) shows a sample mission space with target locations and Fog 3.4(b) shows the value of $R(w,t)$ at a specific time t.

Moreover, following (3·7),

$$P(w, \mathbf{s}(t)) = \sum_{j=1}^{N} \|s_j(t) - w\|^2 \tag{3·47}$$

Using these definitions we introduce a new objective function metric which is added to the objective function in (4·22):

$$J_2(t) = E\Big[\int_{\mathcal{C}} P(w, \mathbf{s}(t)) R(w,t) dw\Big] \tag{3·48}$$

The expectation is a result of $P(w, \mathbf{s}(t))$ and $R(w,t)$ being random variables defined on the same probability space as $x_i(t)$.

Proposition 3.2. *For $R(w,t)$ in (3·46), there exist $c_i > 0$, $i = 1, \ldots, M$, such that:*

$$\int_{\mathcal{C}} R(w,t) dw = \sum_{i=1}^{M} c_i x_i(t) \tag{3·49}$$

Proof. We have

$$\begin{aligned} \int_{\mathcal{C}} R(w,t) &= \int_{\mathcal{C}} \sum_{i=1}^{M} \frac{\alpha_i x_i(t)}{d_i^+(w)} dw \\ &= \sum_{i=1}^{M} \alpha_i \int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)} dw \end{aligned} \tag{3·50}$$

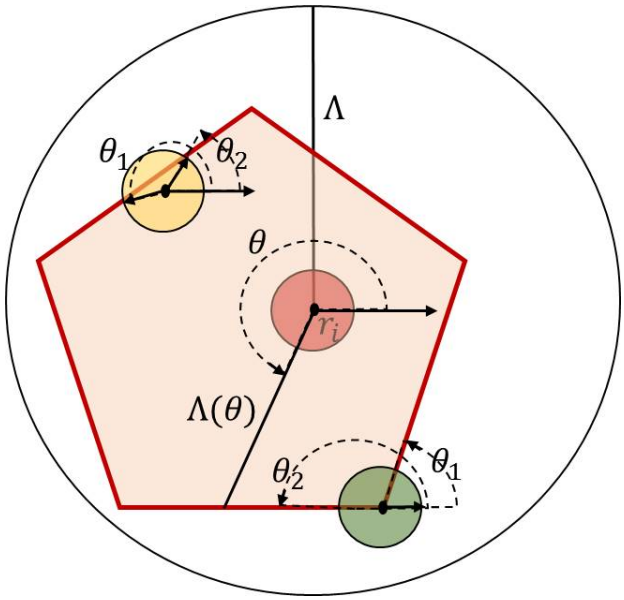


Figure 3·3: One Target $R(w,t)$ Calculation

We now need to find the value of $\int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)}$ for each target i. To do this we first look at the case of one target in a $2D$ space and for now we replace $\mathcal{C}$ with a disk with radius Λ around the target (black circle with radius Λ in Figure 3·3). We can now calculate the above integral for this target using the polar coordinates:

$$\begin{aligned}
&\int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)} dw = \int_0^{2\pi} \int_0^{\Lambda} \frac{x_i(t)}{\max(r_i, r)} dr d\theta \\
&= \int_0^{2\pi} \int_0^{r_i} \frac{x_i(t)}{r_i} dr d\theta + \int_0^{2\pi} \int_{r_i}^{\Lambda} \frac{x_i(t)}{r} dr d\theta \\
&= x_i(t)[2\pi(1 + \log(\frac{\Lambda}{r_i}))]
\end{aligned} \tag{3·51}$$

However in reality, $\mathcal{C}$ is the convex hull of all targets. We will use the same idea to calculate the $\int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)} dw$ for the case of actual convex hull. To do this we consider three separate cases for the target location. In the following

1. Target i and $C(w_i)$ are completely in the interior of $\mathcal{C}$:

This is shown in Figure 3·3 for the target filled in red color. Using the same polar coordinate for each θ we define $\Lambda(\theta)$ to be the distance of the target to the edge of $\mathcal{C}$

in the direction of θ. ($\mathcal{C}$ shown by a filled red polygon in Figure 3·3).

$$\begin{aligned}\int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)} dw &= \int_0^{2\pi} \int_0^{\Lambda(\theta)} \frac{x_i(t)}{d_i^+(r,\theta)} dr d\theta \\ &= \int_0^{2\pi} \int_0^{r_i} \frac{x_i(t)}{r_i} dr d\theta + \int_0^{2\pi} \int_{r_i}^{\Lambda(\theta)} \frac{x_i(t)}{r} dr d\theta \\ &= x_i(t)[2\pi + \int_0^{2\pi} \log(\frac{\Lambda(\theta)}{r_i}) d\theta]\end{aligned} \tag{3·52}$$

The second part in (3·54) has to be calculated knowing $\Lambda(\theta)$ but since we assumed the target is inside the convex hull we know $\Lambda(\theta) \geq r_i$. This means $\log(\frac{\Lambda(\theta)}{r_i}) > 0$ and the $x_i(t)$'s multiplier is a positive value. We can define c_i in (3·49) as:

$$c_i = \alpha_i[2\pi + \int_0^{2\pi} \log(\frac{\Lambda(\theta)}{r_i}) d\theta] \tag{3·53}$$

2. Target i is on the edge of $\mathcal{C}$:

This is shown in Figure 3·3 for the target filled in green color. For this target, we need to do the integration for the appropriate limits of the integral since not all the $C(w_i)$ is inside the $\mathcal{C}$.

$$\begin{aligned}\int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)} dw &= \int_0^{2\pi} \int_0^{\Lambda(\theta)} \frac{x_i(t)}{d_i^+(r,\theta)} dr d\theta \\ &= \int_{\theta_1}^{\theta_2} \int_0^{r_i} \frac{x_i(t)}{r_i} dr d\theta + \int_{\theta_1}^{\theta_2} \int_{r_i}^{\Lambda(\theta)} \frac{x_i(t)}{r} dr d\theta \\ &= x_i(t)[\theta_2 - \theta_1 + \int_{\theta_1}^{\theta_2} \log(\frac{\Lambda(\theta)}{r_i}) d\theta]\end{aligned} \tag{3·54}$$

The second part in (3·54) has to be calculated knowing $\Lambda(\theta)$ but for $\theta \in [\theta_1\ \theta_2]$ we know $\Lambda(\theta) \geq r_i$. This means $\log(\frac{\Lambda(\theta)}{r_i}) > 0$ and by definition $\theta_2 - \theta_1 > 0$ so the $x_i(t)$'s multiplier is a positive value. We can define c_i in (3·49) as:

$$c_i = \alpha_i[\theta_2 - \theta_1 + \int_{\theta_1}^{\theta_2} \log(\frac{\Lambda(\theta)}{r_i}) d\theta] \tag{3·55}$$

3. Target i is in the interior of $\mathcal{C}$ but $C(w_i)$ is not completely inside $\mathcal{C}$:

As a sample the target filled with yellow color in Figure 3·3 falls in this case. For this target, we need to do the integration for the appropriate limits of the integral

since not all the $C(w_i)$ is inside the $\mathcal{C}$.

$$\begin{aligned}\int_{\mathcal{C}} \frac{x_i(t)}{d_i^+(w)} dw &= \int_0^{2\pi} \int_0^{\Lambda(\theta)} \frac{x_i(t)}{d_i^+(r,\theta)} dr d\theta \\ &= \int_0^{2\pi} \int_0^{r(\theta)} \frac{x_i(t)}{r_i} dr d\theta + \int_{\theta_1}^{\theta_2} \int_{r_i}^{\Lambda(\theta)} \frac{x_i(t)}{r} dr d\theta \\ &= x_i(t)[\int_0^{2\pi} \frac{r(\theta)}{r_i} d\theta + \int_{\theta_1}^{\theta_2} \log(\frac{\Lambda(\theta)}{r_i}) d\theta]\end{aligned} \tag{3·56}$$

The $\Lambda(\theta)$ is illustrated the same as the previous case but is only defined for the values of $\theta \in [\theta_1\ \theta_2]$. The two angles are illustrated in Figure 3·3 for the target in yellow. Again since $\Lambda(\theta)$ for the above θ values is greater than r_i so $\log(\frac{\Lambda(\theta)}{r_i}) > 0$ and we know $r(\theta) > 0$ so the $x_i(t)$'s multiplier is a positive value. We can define c_i in (3·49) as:

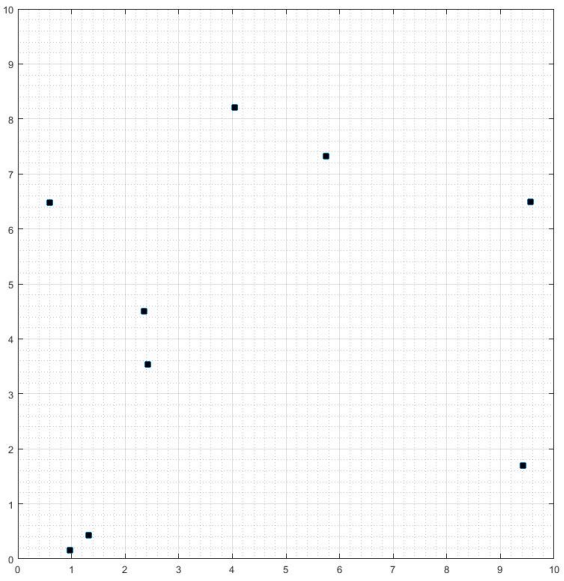
$$c_i = \alpha_i[\int_0^{2\pi} \frac{r(\theta)}{r_i} d\theta + \int_{\theta_1}^{\theta_2} \log(\frac{\Lambda(\theta)}{r_i}) d\theta] \tag{3·57}$$

□

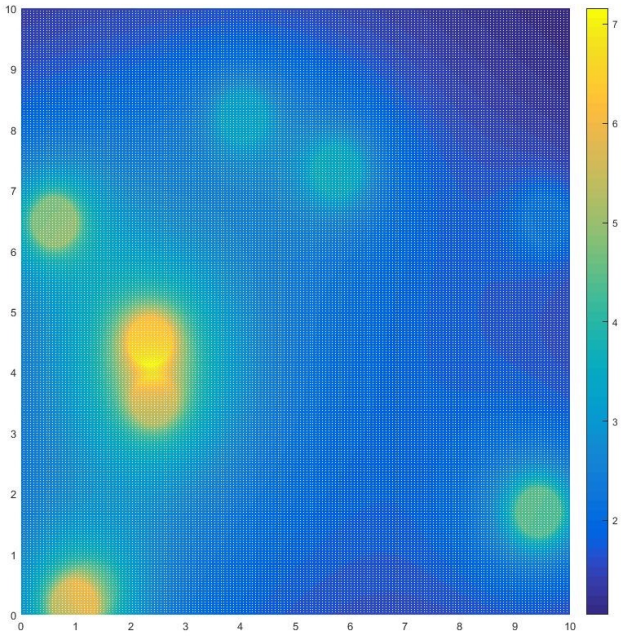
The significance of $J_2(t)$ is that it accounts for the movement of agents through $P(w, \mathbf{s}(t))$ and captures the target state values through $R(w,t)$. Introducing this term in the objective function in the following creates a non-zero gradient even if the agent trajectories are not passing through any targets. We now combine the two metrics in (3·25) and (3·48) and define problem **P3.2**:

$$\mathbf{P3.2}: \qquad \min_{\mathbf{u}(t),\theta(t)} J(T) = \frac{1}{T} \int_0^T [J_1(t) + J_2(t)] dt \tag{3·58}$$

In this problem, the second term is responsible for adjusting the trajectories towards the targets by creating a potential field, while the first term is the original performance metric which is responsible for adjusting the trajectories so as to maximize the data collected once an agent is within a target's sensing range. It can be easily shown that the results in (3·38) hold for problem **P3.2** as well, through the same Hamiltonian analysis presented previously. When $s_j(t)$ follows the parametric functions in (3·41),



(a) Mission Space with dots as target locations



(b) R Function at a sample time t

Figure 3·4: R function illustration

the new metric simply becomes a function of the parameter vector Θ and we have:

$$\min_{\Theta \in F_\Theta} J(\Theta, T) = \frac{1}{T} \int_0^T [J_1(\Theta, t) + J_2(\Theta, t)] dt \tag{3·59}$$

The new objective function's derivative follows the same procedure that was described previously. The first part's derivative can be calculated from (3·43). For the second part we have:

$$\begin{aligned} &\frac{d}{d\Theta} \int_{\tau_k}^{\tau_{k+1}} \int_{\mathcal{C}} P(w, \Theta, t) R(w, \Theta, t) dw dt \\ &\quad = \int_{\tau_k}^{\tau_{k+1}} \int_{\mathcal{C}} \Big[\frac{dP(w, \Theta, t)}{d\Theta} R(w, \Theta, t) + P(w, \Theta, t) \frac{dR(w, \Theta, t)}{d\Theta} \Big] dw dt \end{aligned} \tag{3·60}$$

In the previous section, we raised the problem of no events being excited in a sample realization, in which case the total derivative in (3·43) is zero and the algorithm in (3·12) stalls. Now, looking at (3·60) we can see that if no events occur the second part in the integration which involves $\frac{dR(w,\Theta,t)}{d\Theta}$ will be zero, since $\sum_{i=1}^{M} x_i'(t) = 0$ at all t. However, the first part in the integral does not depend on the events, but calculates the sensitivity of $P(w, \Theta, t)$ in (3·47) with respect to the parameter Θ. Note that the dependence on Θ comes through the parametric description of $\mathbf{s}(t)$ through (4·35). This term ensures that the algorithm in (3·12) does not stall and adjusts trajectories so as to excite the desired events.

3.4 Simulation Results

We provide some simulation results based on an elliptical parametric description for the trajectories in (4·35). The elliptical trajectory formulation is:

$$\begin{aligned} s_j^x(t) = &\quad A_j + a_j \cos \rho_j(t) \cos \phi_j - b_j \sin \rho_j(t) \sin \phi_j \\ s_j^y(t) = &\quad B_j + a_j \cos \rho_j(t) \sin \phi_j + b_j \sin \rho_j(t) \cos \phi_j \end{aligned} \tag{3·61}$$

Here, $\Theta_j = [A_j, B_j, a_j, b_j, \phi_j]$ where A_j, B_j are the coordinates of the center, a_j and b_j are the major and minor axis respectively while $\phi_j \in [0, \pi)$ is the ellipse orientation which is defined as the angle between the x axis and the major axis of the ellipse. The time-dependent parameter $\rho_j(t)$ is the eccentric anomaly of the ellipse. Since an agent is moving with constant speed of 1 on this trajectory, based on (3·38), we have $\dot{s}_j^x(t)^2 + \dot{s}_j^y(t)^2 = 1$, which gives

$$\begin{aligned}\dot{\rho}_j(t) = \Big[\Big(a\sin\rho_j(t)\cos\phi_j + b_j\cos\rho_j(t)\sin\phi_j\Big)^2 \\ + \Big(a\sin\rho_j(t)\sin\phi_j - b_j\cos\rho_j(t)\cos\phi_j\Big)^2\Big]^{-\frac{1}{2}}\end{aligned} \tag{3·62}$$

The first case we consider is a problem with one agent and seven targets located on a circle, as shown in Figure 3·5. We consider a deterministic case with $\sigma_i(t) = 0.5$ for all i. The other problem parameters are $T = 50$, $\mu_{ij} = 100$, $r_i = 0.2$ and $\alpha_i = 1$. A target's sensing range is denoted with solid black circles with the target location at the center. The blue polygon indicates the convex hull produced by the targets. The direction of motion on a trajectory is shown with the small arrow. Starting with an initial trajectory shown in light blue, the on-line trajectory optimization process converges to the trajectory passing through all targets in an efficient manner (shown in dark solid blue). In contrast, starting with this trajectory - which does not pass through any targets - problem **P3.1** does not converge and the initial trajectory remains unchanged. At the final trajectory, $J_1^* = 0.0859$ and $J^* = 0.2128$. Using the obvious shortest path solution, the actual optimal value for J_1 is 0.0739 that results from moving on the edges of the convex hull (which allows for shorter agent travel times).

In the second case, 7 targets are randomly distributed and two agents are cooperatively collecting the data. The problem parameters are $\sigma_i = 0.5,\ \mu_{ij} = 10,\ r_i = 0.5, \alpha_i = 1,\ T = 50$. The initial trajectories for both agents are shown in light green

and blue respectively. We can see that both agent trajectories converge so as to cover all targets, shown in dark green and blue ellipses. At the final trajectories, $J_1^* = 0.1004$ and $J^* = 0.2979$. Note that we may use these trajectories to initialize the corresponding TPBVP, another potential benefit of this approach. This is a much slower process which ultimately converges to $J_1^* = 0.0991$ and $J^* = 0.2776$.

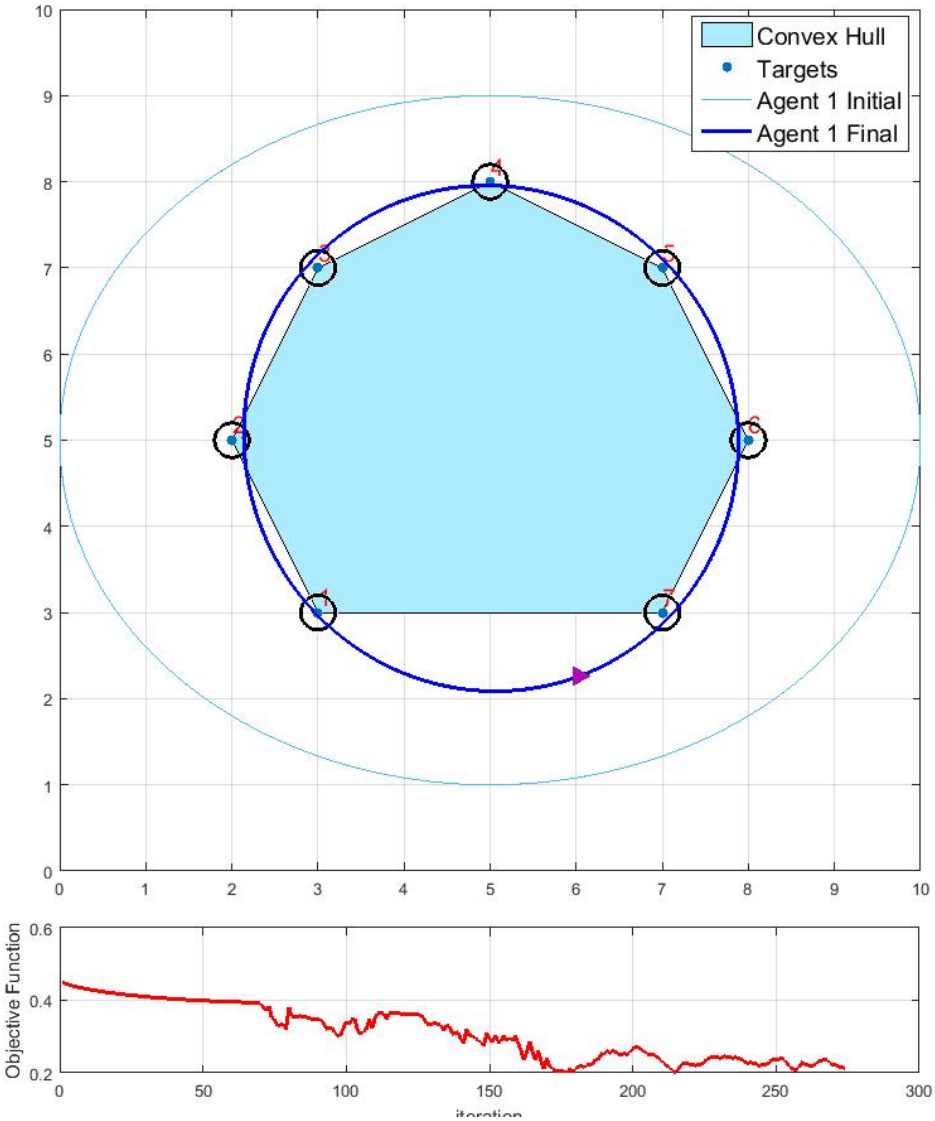


Figure 3·5: One agent and seven target scenario

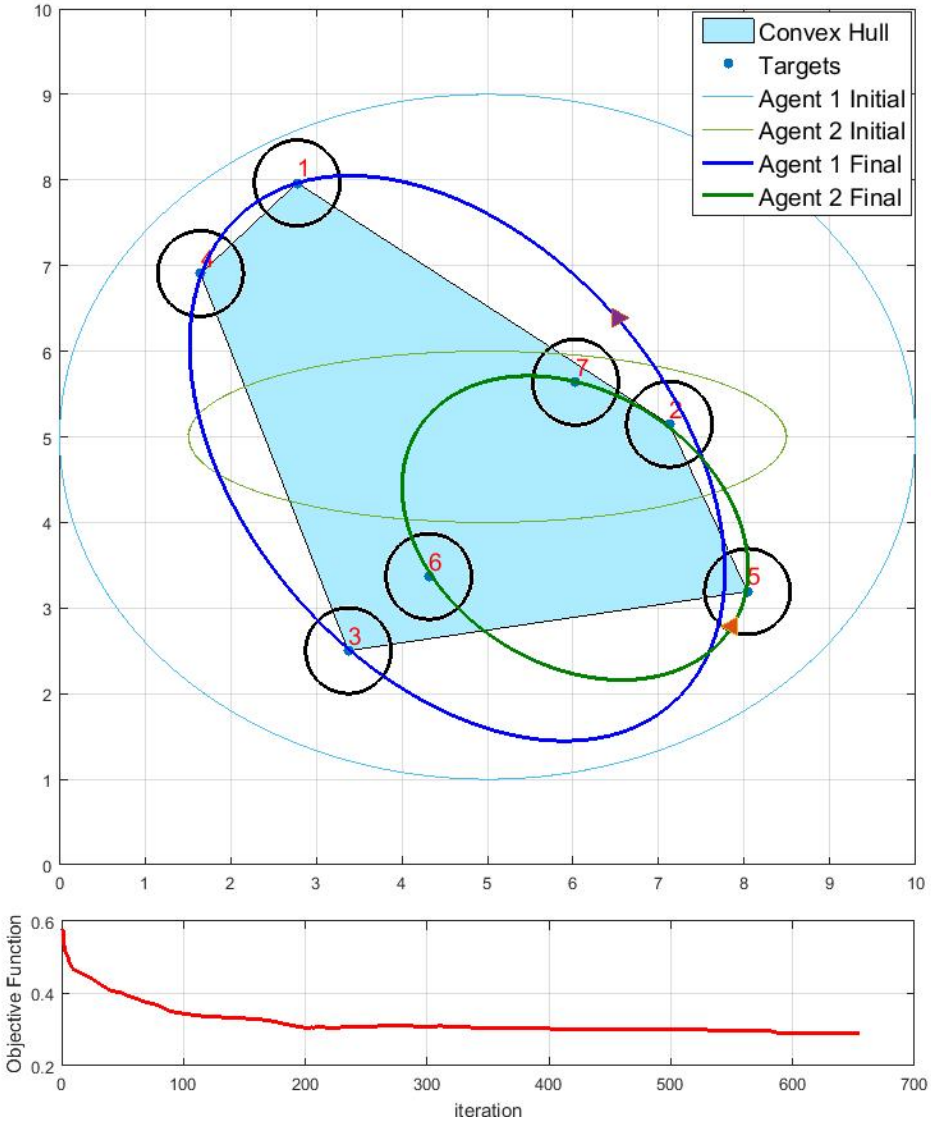


Figure 3·6: Two agent and seven targets scenario

Chapter 4

Data Harvesting Problem

4.1 Problem Formulation

Following what we saw in the previous chapter, we consider a data harvesting problem where N mobile agents collect data from M stationary targets in a two-dimensional mission space S. Each agent may visit one or more of the M targets, collect data from them, and deliver them to a base. It then continues visiting targets, possibly the same as before or new ones, and repeats this process. The objective of the team of agents is to deliver the most amount of data in a fixed time interval T. This problem is more complex than the case of data collection introduced in previous chapter as the data has to delivered to the base in minimum time as well as being collected from the target points.

4.1.1 Queueing Model

The data harvesting problem described above can be viewed as a polling system where mobile agents are serving the targets by collecting data and delivering it to the base. As seen in Figure 4·1, there are three sets of queues. The first set includes the data contents $X_i(t) \in \mathbb{R}^+$ at each target $i = 1, ..., M$ where we use $\sigma_i(t)$ as the instantaneous inflow rate. In general, we treat $\{\sigma_i(t)\}$ as a random process assumed only to be piecewise continuous; we will treat it as a deterministic constant only for the Hamiltonian analysis in the next section. Thus, at time t, $X_i(t)$ is a random variable resulting from the random process $\{\sigma_i(t)\}$.

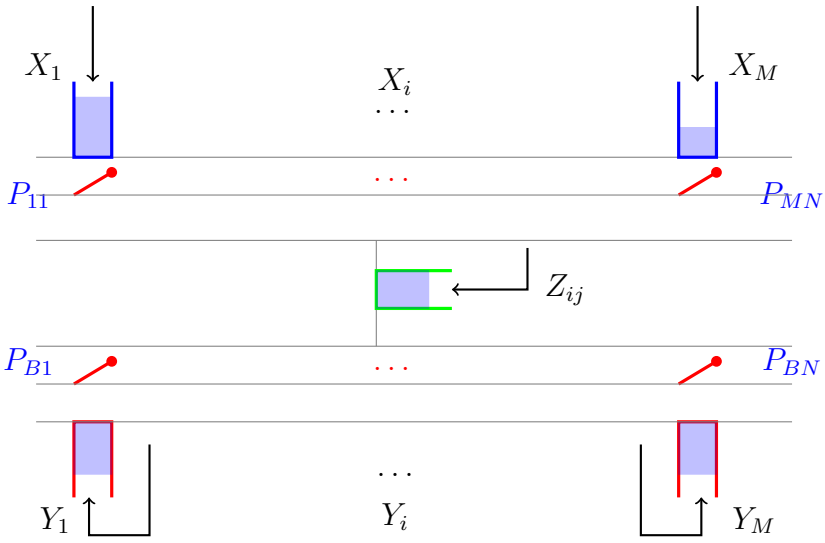


Figure 4·1: Data harvesting queueing model for M targets and N agents

The second set of queues consists of data contents $Z_{ij}(t) \in \mathbb{R}^+$ onboard agent j collected from target i. The last set consists of queues $Y_i(t) \in \mathbb{R}^+$ containing data at the base, one queue for each target, delivered by some agent j. Note that $\{X_i(t)\}$, $\{Z_{ij}(t)\}$ and $\{Y_i(t)\}$ are also random processes.

In Figure 4·1 collection and delivery switches are shown by p_{ij} and $p_{_{Bj}}$. These switches are "on" when agent j is connected to target i or the base respectively. We model the switches by a linear function of the distance to the target and base within their respective finite mutual sensing ranges r_{ij} and $r_{_{Bj}}$.

All queues are modeled as flow systems whose dynamics are given next (however, as we will see, the agent trajectory optimization is driven by events observed in the underlying system where queues contain discrete data packets so that this modeling device has minimal effect on our analysis).

Let $s_j(t) = [s_j^x(t), s_j^y(t)] \in S$ be the position of agent j at time t, Then the state

variable of the system can now be defined as

$$\begin{aligned}\mathbf{X}(t) = [&X_1(t), \ldots, X_M(t), Y_1(t), \ldots, Y_M(t), \\ &Z_{11}(t), \ldots, Z_{MN}(t), s_1^x(t), s_1^y(t), \ldots, s_N^x(t), s_N^y(t)]\end{aligned} \tag{4·1}$$

The position of the agent follows single integrator dynamics at all time:

$$\dot{s}_j^x(t) = u_j(t)\cos\theta_j(t), \qquad \dot{s}_j^y(t) = u_j(t)\sin\theta_j(t) \tag{4·2}$$

$$s_j^x(0) = X_B \qquad s_j^y(0) = Y_B, \quad \forall j$$

where $u_j(t)$ is the scalar speed of the agent (normalized so that $0 \leq u_j(t) \leq 1$), $0 \leq \theta_j(t) < 2\pi$ is the angle relative to the positive direction and $[X_B, Y_B]$ is the location of the base. Thus, we assume that the agent controls its orientation and speed. Note that the agent states $\{s_j(t)\}$, $j = 1, \ldots, N$, are also random processes since the controls are generally dependent on the random queue states. Thus, we ensure that all random processes are defined on a common probability space.

An agent is represented as a particle, so that we will omit the need for any collision avoidance control. The agent dynamics above could be more complicated without affecting the essence of our analysis, but we will limit ourselves here to (4·2).

Following the same idea as in the collection problem in previous chapter, for this specific data harvesting problem, we consider a set of data sources as points $w_i \in S$, $i = 1, \ldots, M$, with associated ranges r_{ij}, so that agent j can collect data from w_i only if the Euclidean distance $d_{ij}(t) = \|w_i - s_j(t)\|$ satisfies $d_{ij}(t) \leq r_{ij}$. This means the $\mathcal{C}(w_i)$ d that was introduced in previous chapter is assumed to be a disk with radius r_{ij}. Similarly, the base is at $w_{_B} = [X_B, Y_B] \in S$ which receives all data collected by the agents. An agent can only deliver data to the base if the Euclidean distance $d_{_{Bj}}(t) = \|w_{_{Bj}} - s_j(t)\|$ satisfies $d_{_{Bj}}(t) \leq r_{Bj}$. Using a function $p : S \times S \to [0, 1]$, we

define the function $p_{ij}(t)$ representing the collection switches in Figure 4·1 as:

$$p_{ij}(t) = p(w_i, s_j(t)) \tag{4·3}$$

$p_{ij}(t)$ is viewed the same way defined in (3·23) with all conditions **A1** through **A4** hold. Similarly, we define:

$$p_{B_j}(t) = p(w_{_B}, s_j(t)) \tag{4·4}$$

As described before, the maximum rate of data collection from target i by agent j is μ_{ij} and the instantaneous rate is $\mu_{ij}p_{ij}(t)$ if j is connected to i.

Now we can define the rest of state dynamics. Dynamics of $X_i(t)$, assuming that agent j is connected to it, are the same as defined in (3·23):

$$\dot{X}_i(t) = \begin{cases} 0 & \text{if } X_i(t) = 0 \text{ and } \sigma_i(t) \leq \mu_{ij}p_{ij}(t) \\ \sigma_i(t) - \mu_{ij}p_{ij}(t) & \text{otherwise} \end{cases} \tag{4·5}$$

Obviously, $\dot{X}_i(t) = \sigma_i(t)$ if $p_{ij}(t) = 0$ for all $j = 1, \ldots, N$.
In order to express the dynamics of $Z_{ij}(t)$, let

$$\tilde{\mu}_{ij}(t) = \begin{cases} \min\left(\frac{\sigma_i(t)}{p_{ij}(t)}, \mu_{ij}\right) & \text{if } X_i(t) = 0 \text{ and } p_{ij}(t) > 0 \\ \mu_{ij} & \text{otherwise} \end{cases} \tag{4·6}$$

This gives us the dynamics as the following:

$$\dot{Z}_{ij}(t) = \begin{cases} 0 & \text{if } Z_{ij}(t) = 0 \text{ and } \tilde{\mu}_{ij}(t)p_{ij}(t) - \beta_{ij}p_{_{B_j}}(t) \leq 0 \\ \tilde{\mu}_{ij}(t)p_{ij}(t) - \beta_{ij}p_{_{B_j}}(t) & \text{otherwise} \end{cases} \tag{4·7}$$

where β_{ij} is the maximum rate of data from target i delivered by agent j. For

simplicity, we assume that: (**A5**) $\|w_i - w_B\| > r_{ij} + r_{Bj}$ for all $i = 1, \ldots, M$ and $j = 1, \ldots, N$, i.e., the agent cannot collect and deliver data at the same time. Therefore, in (4·7) it is always the case that for all i and j, $p_{ij}(t)p_{_{Bj}}(t) = 0$.

Finally, dynamics of $Y_i(t)$ depend on $Z_{ij}(t)$, the content of the on-board queue of each agent j from target i as long as $p_{_{Bj}}(t) > 0$. We define

$$\beta_i(t) = \sum_{j=1}^{N} \beta_{ij} p_{_{Bj}}(t) \mathbf{1}[Z_{ij}(t) > 0] \tag{4·8}$$

to be the total instantaneous delivery rate for target i data, so that the dynamics of $Y_i(t)$ are:

$$\dot{Y}_i(t) = \beta_i(t) \tag{4·9}$$

4.1.2 The Hybrid System

Taking into account the state vector in (4·1) and the dynamics in (4·2), (4·5), (4·7) and (4·9), the data harvesting process is a stochastic hybrid system. Discrete modes of the system are agents visiting a target, agents visiting a base and agents moving not connected to any target or base. The dynamics of agent's state is not changing in this system however, it is possible to add other modes for the agents such as stopping at targets or base. Let's define two distance functions as below:

$$d_{ij}^{+}(t) = \max(0, d_{ij}(t) - r_{ij}),\ d_{_{Bj}}^{+}(t) = \max(0, d_{_{Bj}}(t) - r_{_{Bj}}) \tag{4·10}$$

The above parameters are zero if the agent j is at target i or the base respectively. Now following the previous for type of events that we say in previous chapter for a simpler system, we define a total of seven event types for the data harvesting system. These events are listed in Table 4.1 (the superscript 0 denotes events causing a variable to reach a value of zero from above and the superscript + denotes events causing a variable to become strictly positive from a zero value).

Table 4.1: Hybrid System Events

Event Name	Description
1. ξ_i^0	$X_i(t)$ hits 0, for $i = 1, \ldots, M$
2. ξ_i^+	$X_i(t)$ leaves 0, for $i = 1, \ldots, M$.
3. ζ_{ij}^0	$Z_{ij}(t)$ hits 0, for $i = 1, \ldots, M$, $j = 1, \ldots, N$
4. δ_{ij}^+	$d_{ij}^+(t)$ leaves 0, for $i = 1, \ldots, M$, $j = 1, \ldots, N$
5. δ_{ij}^0	$d_{ij}^+(t)$ hits 0, for $i = 1, \ldots, M$, $j = 1, \ldots, N$
6. Δ_j^+	$d_{Bj}^+(t)$ leaves 0, for $j = 1, \ldots, N$
7. Δ_j^0	$d_{Bj}^+(t)$ hits 0, for $j = 1, \ldots, N$

Observe that each of these events causes a change in at least one of the state dynamics in (4·5), (4·7), (4·9). For example, ξ_i^0 causes a switch in (4·5) from $\dot{X}_i(t) = \sigma_i(t) - \mu_{ij} p_{ij}(t)$ to $\dot{X}_i(t) = 0$. Also note that we have omitted an event ζ_{ij}^+ for $Z_{ij}(t)$ leaving 0 since this event is immediately induced by δ_{ij}^0 when agent j comes within range of target i and starts collecting data causing $Z_{ij}(t)$ to become positive if $Z_{ij}(t) = 0$ and $X_i(t) > 0$. Finally, note that all events above are directly observable during the execution of any agent trajectory and they do not depend on our model of flow queues. For example, if $X_i(t)$ becomes zero, this defines event ξ_i^0 regardless of whether the corresponding queue is based on a flow or on discrete data packets; this observation is very useful in the sequel. A high level hybrid automaton is presented in Figure 4·2 from the point of view of one target i and one agent j. This automaton becomes much more complicated once more targets and agents are to be included.

4.1.3 Performance Measure

Our objective is to maintain minimal data content at all target queues, while maximizing the contents of the delivered data at the base queues. Thus, we define $J_1(t)$ to be the weighted sum of expected target queues content (recalling that $\{\sigma_i(t)\}$ are

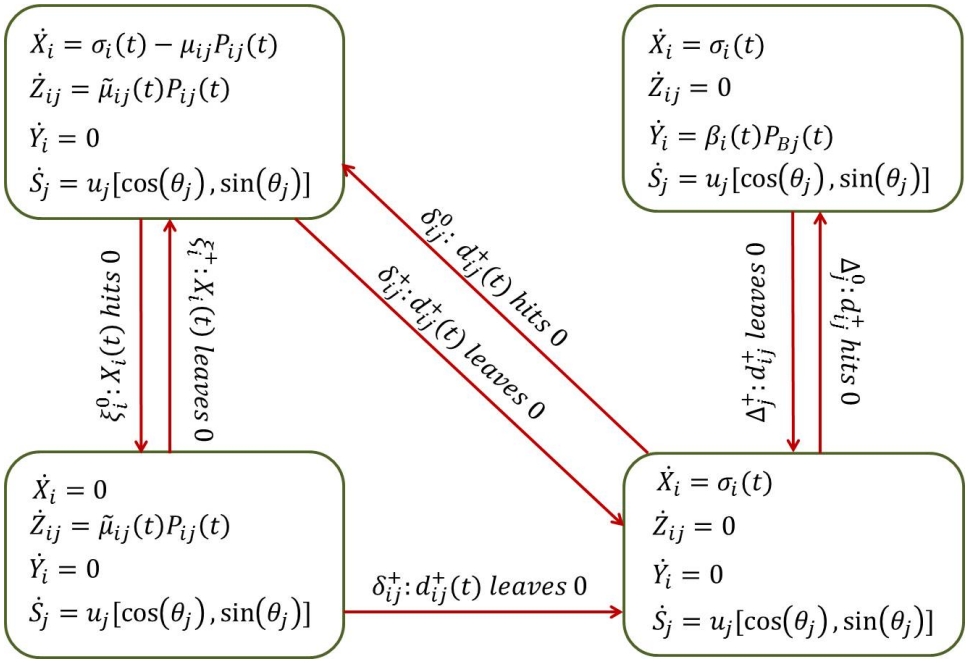


Figure 4·2: One target i and one agent j hybrid automaton

random processes):

$$J_1(t) = \frac{1}{M_X} E[\sum_{i=1}^{M} \alpha_i X_i(t)] \tag{4·11}$$

where the weight α_i represents the importance factor of target i and M_X is a normalizing factor. As you notice this is exactly as we defined J_1 in (3·24). Here, unlike the previous chapter, the data queues are not only at the targets so similarly, we define a weighted sum of expected base queue contents:

$$J_2(t) = \frac{1}{M_Y} E[\sum_{i=1}^{M} \alpha_i Y_i(t)] \tag{4·12}$$

Here M_Y again is a normalizing factor. Therefore, our optimization objective may be a convex combination of (4·11) and (4·12). We set to define the problem **P4.1** as

$$\textbf{P4.1}: \quad \min_{\mathbf{u}(t),\theta(t)} J(T) = \frac{1}{T}\int_0^T \Big(qJ_1(t) - (1-q)J_2(t)\Big)dt \tag{4·13}$$

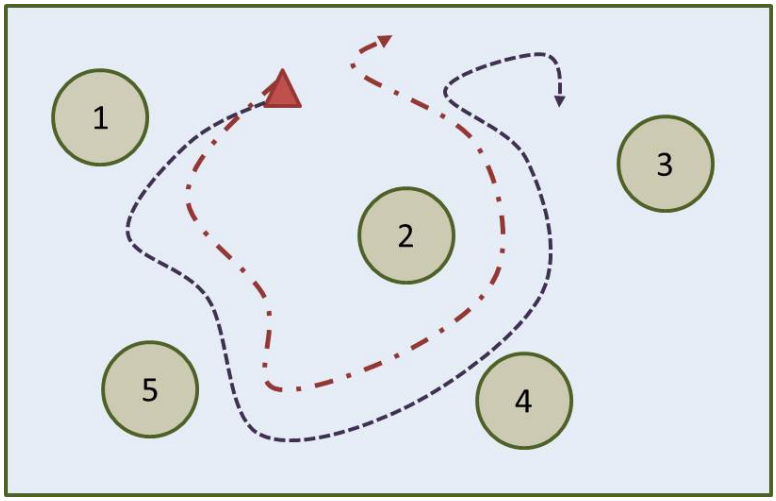


Figure 4·3: Two trajectories with same objective function value

Similar to the collection problem in previous chapter, due to the topology of the data harvesting problem in which only a finite number of targets exist in the mission space (with finite size sensing range), for many trajectories the value of objective function is constant. This means we will see no gradient in the objective function. Simply, if a trajectory does not pass through any target means the system is always in the mode where agents are moving without being connected to any target or base and the value of J is calculated as following:

$$J(T) = \frac{q}{M_X T} \int_0^T t\sigma_i(t)dt \tag{4·14}$$

This is illustrated in Figure 4·3 where two different trajectories are shown for the agent. The blue and red trajectories shown in dotted curves pass through non of the targets resulting in the same objective value for both. In general an infinite number of these trajectories can be found and this large plateau in the objective function creates real problem for the gradient based methods. Next we discuss two nobel additions to the objective function that allow us to apply gradient based methods to such problems.

4.1.4 Agent's Utilization

We need to ensure that the agents are controlled so as to maximize their utilization, i.e., the fraction of time spent performing a useful task by being within range of a target or the base. Equivalently, we aim to minimize the non-productive idling time of each agent during which it is not visiting any target or the base.

The idling time for agent j occurs when $d_{ij}^{+}(t) > 0$ for all i and $d_{Bj}^{+}(t) > 0$. We define the idling function $I_j(t)$:

$$I_j(t) = \log\left(1 + d_{Bj}^{+}(t)\prod_{i=1}^{M} d_{ij}^{+}(t)\right) \tag{4·15}$$

This function has the following properties. First, $I_j(t) = 0$ if and only if the product term inside the bracket is zero, i.e., agent j is visiting a target or the base; otherwise, $I_j(t) > 0$. Second, $I_j(t)$ is monotonically nondecreasing in the number of targets M. The logarithmic function is selected so as to prevent the value of $I_j(t)$ from dominating those of $J_1(\cdot)$ and $J_2(\cdot)$ when included in a single objective function. We define:

$$J_3(t) = \frac{1}{M_I} E[\sum_{j=1}^{N} I_j(t)] \tag{4·16}$$

where M_I is a weight for the idling time effect relative to $J_1(\cdot)$ and $J_2(\cdot)$. Note that $I_j(t)$ is also a random variable since it is a function of the agent states $s_j(t)$, $j = 1, \ldots, N$.

4.1.5 Event Excitation

Following what we introduced in Section 3.3.1, we use the same definition for $R(w,t)$:

$$R(w,t) = \sum_{i=1}^{M} \frac{\alpha_i X_i(t)}{d_i^{+}(w)} \tag{4·17}$$

where $w \in S$, $d_i^+(w) = \max(\|w_i - w\|, r_i)$ and $r_i = \min_j r_{ij}$. Extending the same definition to capture the effect of the base in the problem, for each agent we define a function $R_{Bj}(w,t)$ that generates a density of the amount of data onboard agent j as the following: Also using the same logic we define

$$R_{_{Bj}}(w,t) = \frac{\sum_{i=1}^{M} \alpha_i Z_{ij}(t)}{d_{_B}^+(w)} \tag{4·18}$$

where $d_{_B}^+(w) = \max(\|w_{_B} - w\|, r_{_B})$ is a constant and $r_{_B} = \min_j r_{_{Bj}}$. The intuition behind (4·18) is that if agent j is carrying data meaning $\sum_{i=1}^{M} Z_{ij}(t) > 0$ then it feels a higher density around the base. In other words, we create a fictitious target point at the base for agents that are carrying data.

Now following the (3·47), for each agent j we define:

$$P_j(w,t) = \|s_j(t) - w\|^2 \tag{4·19}$$

which again provides the interaction between agent j and the environment in terms of a quadratic travel cost for from agent j's position s_j to each point w. Finally, similar to (3·48) we have:

$$J_4(t) = \frac{1}{M_R} E\Big[\sum_{j=1}^{N} \int_{\mathcal{C}} \Big(R(w,t) + R_{_{Bj}}(w,t)\Big) P_j(w,t) dw\Big] \tag{4·20}$$

Where M_R is a normalizing factor and $\mathcal{C}$ is the convex hull built by targets and the base.

4.1.6 Final Cost

Finally, we define a terminal cost at T capturing the expected value of the amount of data left on board the agents, noting that the effect of this term vanishes as T goes

to infinity as long as all $E[Z_{ij}(T)]$ remain bounded:

$$J_f(T) = \frac{1}{TM_Z} E\Big[\sum_{i=1}^{M}\sum_{j=1}^{N} \alpha_i Z_{ij}(T)\Big] \tag{4·21}$$

where M_Z again is a normalizing factor. We might not take into account this final cost when the trajectories are considered to be periodically used. For simplicity, we will in the sequel assume that $\alpha_i = 1$ for all i.

4.1.7 Optimization Problem

We can now formulate a stochastic optimization problem **P4.2** where the control variables are the agent speeds and headings denoted by the vectors $\mathbf{u}(t) = [u_1(t), \ldots, u_N(t)]$ and $\theta(t) = [\theta_1(t), \ldots, \theta_N(t)]$ respectively (omitting their dependence on the full system state at t). We combine the objective function components in (4·16), (4·20) and (4·21) to obtain:

$$\mathbf{P4.2}: \quad \min_{\mathbf{u}(t),\theta(t)} J(T) = \frac{1}{T}\int_0^T \Big(qJ_1(t) - (1-q)J_2(t) + J_3(t) + J_4(t)\Big)dt + J_f(T) \tag{4·22}$$

where $q \in [0,1]$ is a weight capturing the relative importance of collected data as opposed to delivered data. We have used $q = 0.5$ throughout this study. Also $0 \leq u_j(t) \leq 1$ and $0 \leq \theta_j(t) < 2\pi$.

Normalizing Factors

We introduced five normalizing factors M_X, M_Y, M_I, M_R and M_Z. This normalization ensures that all different segments have similar forces in minimizing the value of the objective function. We use an upper bound for the value of each segment for the

normalizing factor.

$$\begin{aligned} M_X &= M_Y = M_Z = T\sum_i \sigma_i(0) \\ M_I &= \log\left(1+\sqrt{L_1^2+L_2^2}^{M+1}\right) \\ M_R &= \frac{TL_1L_2(L_1^2+L_2^2)}{r}\sum_i \sigma_i(0), r = \frac{\sum_i r_i}{M} \end{aligned} \tag{4·23}$$

Using these normalizing factors we can observe that the unattainable minimum of the total objective function is -0.5 which is if $J_1 = J_3 = J_4 = 0$ and J_2 is at its maximum of 1 with $q = 0.5$. This value is obviously never attained.

4.2 Optimization Methodology

In this section, we address **P4.2** in a setting where all data arrival processes are deterministic, so that all expectations in (4·11)-(4·21) degenerate to their arguments. On a completely similar path with the previous chapter we proceed with a standard Hamiltonian analysis leading to a Two Point Boundary Value Problem (TPBVP) where the states and costates are known at $t = 0$ and $t = T$ respectively. We define the associated costate vector to (4·1):

$$\begin{aligned} \boldsymbol{\lambda}(t) = [&\lambda_1(t),\ldots,\lambda_M(t),\gamma_1(t),\ldots,\gamma_M(t),\phi_{11}(t),\ldots,\phi_{MN}(t), \\ &\eta_1^x(t),\eta_1^y(t),\ldots,\eta_N^x(t),\eta_N^y(t)] \end{aligned} \tag{4·24}$$

The Hamiltonian is

$$\begin{aligned} H(\mathbf{X},\boldsymbol{\lambda},\mathbf{u},\theta) &= \frac{1}{T}\Big[qJ_1(t)-(1-q)J_2(t)+J_3(t)+J_4(t)\Big] \\ &+\sum_i \lambda_i(t)\dot{X}_i(t)+\sum_i \gamma_i(t)\dot{Y}_i(t)+\sum_i\sum_j \phi_{ij}(t)\dot{Z}_{ij}(t) \\ &+\sum_j \left(\eta_j^x(t)u_j(t)\cos\theta_j(t)+\eta_j^y(t)u_j(t)\sin\theta_j(t)\right) \end{aligned} \tag{4·25}$$

where the costate equations are

$$\begin{aligned}
\dot{\lambda}_i(t) &= -\frac{\partial H}{\partial X_i} = -\frac{1}{T}[\frac{q}{M_X} + \frac{1}{M_R}\sum_j \int_S \frac{\alpha_i P_j(w,t)}{d_i^+(w)} dw] \;\; \lambda_i(T) = 0 \\
\dot{\gamma}_i(t) &= -\frac{\partial H}{\partial Y_i} = \frac{1-q}{TM_Y} \quad \gamma_i(T) = 0 \\
\dot{\phi}_{ij}(t) &= -\frac{\partial H}{\partial Z_{ij}} = -\frac{1}{TM_R}\int_S \frac{\alpha_i P_j(w,t)}{d_B^+(w)} dw \quad \phi_{ij}(T) = \frac{\partial J_f}{\partial Z_{ij}}\Big|_T
\end{aligned} \tag{4·26}$$

$$\begin{aligned}
\dot{\eta}_j^x(t) &= -\frac{\partial H}{\partial s_j^x} \\
&= -\left[\frac{1}{TM_I}\frac{\partial I_j(t)}{\partial s_j^x} + \frac{1}{TM_R}\sum_j \int_S (R(w,t) + R_{Bj}(w,t))\frac{\partial P_j(w,t)}{\partial s_j^x} dw \right. \\
&\quad \left. + \sum_i \frac{\partial}{\partial s_j^x}\lambda_i(t)\dot{X}_i(t) + \sum_i \frac{\partial}{\partial s_j^x}\gamma_i(t)\dot{Y}_i(t) + \sum_i \frac{\partial}{\partial s_j^x}\phi_{ij}(t)\dot{Z}_{ij}(t)\right]
\end{aligned} \tag{4·27}$$

$$\begin{aligned}
\dot{\eta}_j^y(t) &= -\frac{\partial H}{\partial s_j^y} \\
&= -\left[\frac{1}{TM_I}\frac{\partial I_j(t)}{\partial s_j^y} + \frac{1}{TM_R}\sum_j \int_S (R(w,t) + R_{Bj}(w,t))\frac{\partial P_j(w,t)}{\partial s_j^y} dw \right. \\
&\quad \left. + \sum_i \frac{\partial}{\partial s_j^y}\lambda_i(t)\dot{X}_i(t) + \sum_i \frac{\partial}{\partial s_j^y}\gamma_i(t)\dot{Y}_i(t) + \sum_i \frac{\partial}{\partial s_j^y}\phi_{ij}(t)\dot{Z}_{ij}(t)\right]
\end{aligned} \tag{4·28}$$

$$\eta_j^x(T) = \eta_j^y(T) = 0 \tag{4·29}$$

From (4·25), after some trigonometric manipulations, we get

$$\begin{aligned}
H(\mathbf{X}, \boldsymbol{\lambda}, \mathbf{u}, \theta) &= \frac{1}{T}\Big[qJ_1(t) - (1-q)J_2(t) + J_3(t) + J_4(t)\Big] \\
&+ \sum_i \lambda_i(t)\dot{X}_i(t) + \sum_i \gamma_i(t)\dot{Y}_i(t) + \sum_i \sum_j \phi_{ij}(t)\dot{Z}_{ij}(t) \\
&+ \sum_j u_j(t)\mathrm{sgn}(\eta_j^y(t))\sqrt{\eta_j^x(t)^2 + \eta_j^y(t)^2}\sin(\theta_j(t) + \psi_j(t))
\end{aligned} \tag{4·30}$$

where $\tan\psi_j(t) = \frac{\eta_j^x(t)}{\eta_j^y(t)}$ for $\eta_j^y(t) \neq 0$ and $\psi_j(t) = \mathrm{sgn}(\eta_j^x(t))\frac{\pi}{2}$ if $\eta_j^y(t) = 0$. Applying the Pontryagin principle to (4·25) with $(\mathbf{u}^*, \theta^*)$ being the optimal control,

we have:

$$H(\mathbf{X}^*, \boldsymbol{\lambda}^*, \mathbf{u}^*, \theta^*) = \min_{\mathbf{u}(t), \theta(t)} H(\mathbf{X}, \boldsymbol{\lambda}, \mathbf{u}, \theta) \tag{4·31}$$

From (4·30) we easily see that we can always make the $u_j(t)$ multiplier to be negative, hence, recalling that $0 \leq u_j(t) \leq 1$,

$$u_j^*(t) = 1 \tag{4·32}$$

Following the Hamiltonian definition in (4·25) we have:

$$\frac{\partial H}{\partial \theta_j} = -\eta_j^x(t) u_j(t) \sin\theta_j(t) + \eta_j^y(t) u_j(t) \cos\theta_j(t) \tag{4·33}$$

and setting $\frac{\partial H}{\partial \theta_j} = 0$ the optimal heading $\theta_j^*(t)$ should satisfy:

$$\tan\theta_j^*(t) = \frac{\eta_j^y(t)}{\eta_j^x(t)} \tag{4·34}$$

Since $u_j^*(t) = 1$, we only need to evaluate $\theta_j^*(t)$ for all $t \in [0, T]$. This is accomplished by discretizing the problem in time and numerically solving a TPBVP with a forward integration of the state and a backward integration of the costate. Solving this problem, quickly becomes intractable as the number of agents and targets grows.

We again resort to parametric representation of the trajectories and study two general parametric trajectory descriptions.

4.2.1 Agent Trajectory Parameterization

Following the same idea in (3·41) we define:

$$s_j^x(t) = f_x(\Theta_j, \rho_j(t)), \qquad s_j^y(t) = f_y(\Theta_j, \rho_j(t)) \tag{4·35}$$

where the function $\rho_j(t)$ controls the position of the agent on its trajectory at time t and Θ_j is a vector of parameters controlling the shape and location of the agent j

trajectory. Let $\Theta = [\Theta_1, \ldots, \Theta_N]$. We now replace problem **P4.2** in (4·22) by problem **P4.3**:

$$\mathbf{P4.3}: \quad \min_{\Theta \in F_\Theta} \frac{1}{T} \int_0^T \Big[qJ_1(\Theta, t) - (1-q)J_2(\Theta, t) + J_3(\Theta, t) + J_4(\Theta, t) \Big] dt + J_f(\Theta, T) \tag{4·36}$$

where we return to allowing arbitrary stochastic data arrival processes $\{\sigma_i(t)\}$ so that **P4.3** is a parametric stochastic optimization problem with F_Θ appropriately defined depending on (4·35). The cost function in (4·36) is written as

$$J(\Theta, T; \mathbf{X}(\Theta, 0)) = E[\mathcal{L}(\Theta, T; \mathbf{X}(\Theta, 0))] \tag{4·37}$$

where $\mathcal{L}(\Theta, T; \mathbf{X}(\Theta, 0))$ is a sample function defined over $[0, T]$ and $\mathbf{X}(\Theta, 0)$ is the initial value of the state vector. For convenience, in the sequel we will use $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_4$, $\mathcal{L}_f$ to denote sample functions of J_1, J_2, J_3 ,J_4 and J_f respectively. Note that in (4·36) we suppress the dependence of the four objective function components on the controls $\mathbf{u}(t)$ and $\theta(t)$ and stress instead their dependence on the parameter vector Θ. In the rest of this chapter following the simulation results in Chapter 3, we will consider two families of trajectories motivated by a similar approach used in the multi-agent persistent monitoring problem in (Lin and Cassandras, 2015): *elliptical* trajectories and a *Fourier series* trajectory representation which is more general and better suited for non-uniform target topologies. Following the IPA calculus that was introduced in 3.2 we can calculate an unbiased estimate for the derivative of $\mathcal{L}(\Theta, T; \mathbf{X}(\Theta, 0))$ with respect to Θ. As mentioned before, the value of the IPA approach is twofold: (i) The sample gradient $\nabla\mathcal{L}(\Theta, T)$ can be obtained on line based on observable sample path data *only*, and (ii) $\nabla\mathcal{L}(\Theta, T)$ is an unbiased estimate of $\nabla J(\Theta, T)$ under mild technical conditions as shown in (Cassandras et al., 2010). Therefore, we can use

$\nabla\mathcal{L}(\Theta, T)$ in a standard gradient-based stochastic optimization algorithm

$$\Theta^{l+1} = \Theta^l - \boldsymbol{\nu}_l \nabla\mathcal{L}(\Theta^l, T), \quad l = 0, 1, \ldots \tag{4·38}$$

to converge (at least locally) to an optimal parameter vector Θ^* with a proper selection of a step-size sequence $\{\boldsymbol{\nu}_l\}$ (Kushner and Yin, 2003). We emphasize that this process is carried out *on line*, i.e., the gradient is evaluated by observing a trajectory with given Θ over $[0, T]$ and is iteratively adjusting it until convergence is attained.

Objective Function Gradient

The sample function gradient $\nabla\mathcal{L}(\Theta,T)$ needed in (4·38) is obtained from (4·36) assuming a total of K events over $[0\ T]$ with $\tau_{_{K+1}}=T$ and $\tau_0=0$:

$$\begin{aligned}
\nabla\mathcal{L}(\Theta,T;\mathbf{X}(\Theta;0)) &= \frac{1}{T}\nabla\Big[\int_0^T\Big(q\mathcal{L}_1(\Theta,t)-(1-q)\mathcal{L}_2(\Theta,t)+\mathcal{L}_3(\Theta,t)\\
&\quad+\mathcal{L}_4(\Theta,t)\Big)dt\Big]+\nabla\mathcal{L}_f(\Theta,T)\\
&= \frac{1}{T}\nabla\Big[\sum_{k=0}^{K}\int_{\tau_k}^{\tau_{k+1}}\Big(q\mathcal{L}_1(\Theta,t)-(1-q)\mathcal{L}_2(\Theta,t)+\mathcal{L}_3(\Theta,t)\\
&\quad+\mathcal{L}_4(\Theta,t)\Big)dt\Big]+\nabla\mathcal{L}_f(\Theta,T)\\
&= \frac{1}{T}\Big[\sum_{k=0}^{K}q\Big(\int_{\tau_k}^{\tau_{k+1}}\nabla\mathcal{L}_1(\Theta,t)dt+\mathcal{L}_1(\Theta,\tau_{k+1})\tau'_{k+1}-\mathcal{L}_1(\Theta,\tau_k)\tau'_k\Big)\\
&\quad-(1-q)\Big(\int_{\tau_k}^{\tau_{k+1}}\nabla\mathcal{L}_2(\Theta,t)dt+\mathcal{L}_2(\Theta,\tau_{k+1})\tau'_{k+1}-\mathcal{L}_2(\Theta,\tau_k)\tau'_k\Big)\\
&\quad+\Big(\int_{\tau_k}^{\tau_{k+1}}\nabla\mathcal{L}_3(\Theta,t)dt+\mathcal{L}_3(\Theta,\tau_{k+1})\tau'_{k+1}-\mathcal{L}_3(\Theta,\tau_k)\tau'_k\Big)\\
&\quad+\Big(\int_{\tau_k}^{\tau_{k+1}}\nabla\mathcal{L}_4(\Theta,t)dt+\mathcal{L}_4(\Theta,\tau_{k+1})\tau'_{k+1}-\mathcal{L}_4(\Theta,\tau_k)\tau'_k\Big)\Big]\\
&\quad+\nabla\mathcal{L}_f(\Theta,T)\\
&= \frac{1}{T}\Big[\sum_{k=0}^{K}\int_{\tau_k}^{\tau_{k+1}}\Big(q\nabla\mathcal{L}_1(\Theta,t)-(1-q)\nabla\mathcal{L}_2(\Theta,t)+\nabla\mathcal{L}_3(\Theta,t)\\
&\quad+\nabla\mathcal{L}_4(\Theta,t)\Big)dt\Big]+\nabla\mathcal{L}_f(\Theta,T)
\end{aligned}\tag{4·39}$$

The last step follows from the continuity of the state variables which causes adjacent limit terms in the sum to cancel out. Therefore, $\nabla\mathcal{L}(\Theta,T)$ does not have any direct dependence on any τ'_k; this dependence is indirect through the state derivatives involved in the four individual gradient terms. Referring to (4·11), the first term involves $\nabla\mathcal{L}_1(\Theta,t)$ which is as a sum of $X'_i(t)$ derivatives. Similarly, $\nabla\mathcal{L}_2(\Theta,t)$ is a sum of $Y'_i(t)$ derivatives and $\nabla\mathcal{L}_f(\Theta,T)$ requires only $Z'_{ij}(T)$. The third term,

$\nabla\mathcal{L}_3(\Theta,t)$, requires derivatives of $I_j(t)$ in (4·15) which depend on the derivatives of the max function in (4·10) and the agent state derivatives $s_j'(t)$ with respect to Θ. The term $\nabla\mathcal{L}_4(\Theta,t)$ needs the values of $X_i'(t)$ and $Z_{ij}'(t)$. The gradients of the last two terms are derived in the appendix. Possible discontinuities in these derivatives occur when any of the last four events in Table 4.1 takes place.

In summary, the evaluation of (4·39) requires the state derivatives $X_i'(t)$, $Z_{ij}'(t)$, $Y_i'(t)$, and $s_j'(t)$. The latter are easily obtained for any specific choice of f_x and f_y in (4·35) and are shown in Appendix A.2.1. The former require a rather laborious use of (3·16), (3·18) and (3·19).

4.3 IPA Derivatives Calculation

In this section, we derive all event time derivatives and state derivatives with respect to the controllable parameter Θ for each event by applying the IPA equations.

1. **Event** ξ_i^0: This event causes a transition from $X_i(t)>0$, $t<\tau_k$ to $X_i(t)=0$, $t\geq\tau_k$. The switching function is $g_k(\Theta,\mathbf{X})=X_i$ so $\frac{\partial g_k}{\partial X_i}=1$. From (3·19) and (4·5):

$$\begin{aligned}\tau_k' &= -\Big(\frac{\partial g_k}{\partial X_i}f_k(\tau_k^-)\Big)^{-1}\Big(g_k'+\frac{\partial g_k}{\partial X_i}X_i'(\tau_k^-)\Big)\\ &= -\frac{X_i'(\tau_k^-)}{\sigma_i(\tau_k)-\mu_{ij}p_{ij}(\tau_k)}\end{aligned} \tag{4·40}$$

where agent j is the one connected to i at $t=\tau_k$ and we have used the assumption that two events occur at the same time w.p. 0, hence $\sigma_i(\tau_k^-)=\sigma_i(\tau_k)$. From (3·16), (3·18) and (3·19), since $\dot{X}_i(t)=0$, for $\tau_k\leq t<\tau_{k+1}$:

$$\frac{d}{dt}X_i'(t)=\frac{\partial\dot{X}_i(t)}{\partial X_i(t)}X_i'(t)+\dot{X}_i'(t)=0 \tag{4·41}$$

$$\begin{aligned} X_i'(\tau_k^+) &= X_i'(\tau_k^-) + \Big[\Big(\sigma_i(\tau_k) - \mu_{ij}p_{ij}(\tau_k)\Big) - 0\Big]\tau_k{}' \\ &= X_i'(\tau_k^-) - \frac{X_i'(\tau_k^-)\Big(\sigma_i(\tau_k) - \mu_{ij}p_{ij}(\tau_k)\Big)}{\sigma_i(\tau_k) - \mu_{ij}p_{ij}(\tau_k)} = 0 \end{aligned} \tag{4·42}$$

For $X_r(t)$, $r \neq i$, the dynamics of $X_r(t)$ in (4·5) are unaffected and we have:

$$X_r'(\tau_k^+) = X_r'(\tau_k^-) \tag{4·43}$$

If $X_r(\tau_k) > 0$ and agent l is connected to it, then

$$\begin{aligned} \frac{d}{dt}X_r'(t) &= \frac{\partial \dot{X}_r(t)}{\partial X_r(t)}X_r'(t) + \dot{X}_r'(t) \\ &= \sigma_r'(t) - \mu_{rl}p_{rl}'(\tau_k) = -\mu_{rl}p_{rl}'(t) \end{aligned} \tag{4·44}$$

and if $X_r(t) = 0$ in $[\tau_k, \tau_{k+1}]$ or if no agents are connected to i, then $\frac{d}{dt}X_r'(t) = 0$.

For $Y_r(t)$, $r = 1, \ldots, M$, the dynamics of $Y_r(t)$ in (4·9) are not affected by the event ξ_i^0 at τ_k, hence

$$Y_r'(\tau_k^+) = Y_r'(\tau_k^-) \tag{4·45}$$

and since $\dot{Y}_r(t) = \beta_r(t)$, for $\tau_k \leq t < \tau_{k+1}$:

$$\frac{d}{dt}Y_r'(t) = \frac{\partial \dot{Y}_r(t)}{\partial Y_r(t)}Y_r'(t) + \dot{Y}_r'(t) = \beta_r'(t) \tag{4·46}$$

For $Z_{ij}(t)$, we must have $Z_{ij}(\tau_k) > 0$ since $X_i(\tau_k^-) > 0$, hence $\tilde{\mu}_{ij}(\tau_k^-) > 0$ and from (3·16):

$$\begin{aligned} Z_{ij}'(\tau_k^+) &= Z_{ij}'(\tau_k^-) + \Big[\dot{Z}_{ij}(\tau_k^-) - \dot{Z}_{ij}(\tau_k^+)\Big]\tau_k' \\ &= Z_{ij}'(\tau_k^-) + \Big[\tilde{\mu}_{ij}(\tau_k^-) - \tilde{\mu}_{ij}(\tau_k^+)\Big]p_{ij}(\tau_k)\tau_k' \end{aligned} \tag{4·47}$$

Since $X_i(\tau_k^-) > 0$, from (4·6) we have $\tilde{\mu}_{ij}(\tau_k^-) = \mu_{ij}$. At τ_k^+, j remains connected to

target i with $\tilde{\mu}_{ij}(\tau_k^+) = \sigma_i(\tau_k^+)/p_{ij}(\tau_k) = \sigma_i(\tau_k)/p_{ij}(\tau_k)$ and we get

$$\begin{aligned} Z'_{ij}(\tau_k^+) &= Z'_{ij}(\tau_k^-) + \frac{-X'_i(\tau_k^-)\Big[\mu_{ij}p_{ij}(\tau_k) - \sigma_i(\tau_k)\Big]}{\sigma_i(\tau_k) - \mu_{ij}p_{ij}(\tau_k)} \\ &= Z'_{ij}(\tau_k^-) + X'_i(\tau_k^-) \end{aligned} \tag{4·48}$$

From (3·18) for $\tau_k \leq t < \tau_{k+1}$:

$$\begin{aligned} \frac{d}{dt}Z'_{ij}(t) &= \frac{\partial \dot{Z}_{ij}(t)}{\partial Z_{ij}(t)}Z'_{ij}(t) + \dot{Z}'_{ij}(t) \\ &= \dot{Z}'_{ij}(t) = \Big(\tilde{\mu}_{ij}(t)p'_{ij}(t) - \beta_{ij}p'_{B_j}(t)\Big) \end{aligned} \tag{4·49}$$

Since $\tilde{\mu}_{ij}(t) = \sigma_i(t)/p_{ij}(t)$ for the agent which remains connected to target i after this event, it follows that $\frac{\partial}{\partial\Theta}[\tilde{\mu}_{ij}(t)p_{ij}(t)] = 0$. Moreover, $p_{B_j}(t) = 0$ by our assumption that agents cannot be within range of the base and targets at the same time and we get

$$\frac{d}{dt}Z'_{ij}(t) = 0 \tag{4·50}$$

Otherwise, for $r \neq j$, we have $\tilde{\mu}_{ir}(t) = 0$ and we get:

$$\frac{d}{dt}Z'_{ir}(t) = -\beta_{ir}p'_{Br}(t) \tag{4·51}$$

Finally, for $Z_{rj}(t)$, $r \neq i$ we have $Z'_{rj}(\tau_k^+) = Z'_{rj}(\tau_k^-)$. If $Z_{rj}(t) = 0$ in $[\tau_k, \tau_{k+1})$, then $\frac{d}{dt}Z'_{rj}(t) = 0$. Otherwise, we get $\frac{d}{dt}Z'_{rj}(t)$ from (4·49) with i replaced by r.

2. **Event** ξ_i^+: This event causes a transition from $X_i(t) = 0$, $t \leq \tau_k$ to $X_i(t) > 0$, $t > \tau_k$. Note that this transition can occur as an exogenous event when an empty queue $X_i(t)$ gets a new arrival in which case we simply have $\tau'_k = 0$ since the exogenous event is independent of the controllable parameters. In the endogenous case, however, we have the switching function $g_k(\Theta, \mathbf{X}) = \sigma_i(t) - \mu_{ij}p_{ij}(t)$ in which agent j is connected to target i at $t = \tau_k$. Assuming $s'_j(t) = [\frac{\partial s_j^x}{\partial\Theta} \ \frac{\partial s_j^y}{\partial\Theta}]^\top$ and $\dot{s}_j = [\dot{s}_j^x \ \dot{s}_j^y]^\top$,

from (3·19):

$$\tau_k{}' = -\Big(\frac{\partial g_k}{\partial s_j} s_j'(\tau_k)\Big)\Big(g_k' \dot{s}_j(\tau_k)\Big)^{-1} \tag{4·52}$$

At τ_k we have $\sigma_i(\tau_k) = \mu_{ij} p_{ij}(\tau_k)$. Therefore from (3·16):

$$\begin{aligned} X_i'(\tau_k^+) &= X_i'(\tau_k^-) + [\dot{X}_i(\tau_k^-) - \dot{X}_i(\tau_k^+)]\tau_k{}' \\ &= X_i'(\tau_k^-) + \Big(0 - \sigma_i(\tau_k) + \mu_{ij} p_{ij}(\tau_k)\Big)\tau_k{}' = X_i'(\tau_k^-) \end{aligned} \tag{4·53}$$

Having $X_i(t) > 0$ in $[\tau_k, \tau_{k+1})$ we know $\dot{X}_i(t) = \sigma_i(t) - \mu_{ij} p_{ij}(t)$ therefor, we can get $\frac{d}{dt} X_i'(t)$ from (4·44) with r and l replaced by i and j. For $X_r(t)$, $r \neq i$, if $X_r(\tau_k) > 0$ and agent l is connected to r then $\dot{X}_r(\tau_k) = \sigma_r(\tau_k) - \mu_{rl} p_{rl}(\tau_k)$, therefor, we get $X_r'(\tau_k^+)$ from (4·43) while in $[\tau_k, \tau_{k+1})$ we have $\frac{d}{dt} X_r'(t)$ from (4·44). If $X_r(\tau_k) = 0$ or if no agent is connected to target r, $\dot{X}_r(\tau_k) = 0$. Thus, $X_r'(\tau_k^+) = X_r'(\tau_k^-)$ and $\frac{d}{dt} X_r'(t) = 0$. For $Y_r(t)$, $r = 1, \ldots, M$ the dynamics of $Y_r(t)$ in (4·9) are not affected by the event at τ_k hence, we can get $Y_r'(\tau_k^+)$ and $\frac{d}{dt} Y_r'(t)$ in $[\tau_k, \tau_{k+1})$ from (4·45) and (4·46) respectively.

For $Z_{ij}(t)$ assuming agent j is the one connected to target i, we have:

$$\begin{aligned} Z_{ij}'(\tau_k^+) &= Z_{ij}'(\tau_k^-) + \Big[\dot{Z}_{ij}(\tau_k^-) - \dot{Z}_{ij}(\tau_k^+)\Big]\tau_k' \\ &= Z_{ij}'(\tau_k^-) + \Big[\tilde{\mu}_{ij}(\tau_k^-) - \tilde{\mu}_{ij}(\tau_k^+)\Big] p_{ij}(\tau_k)\tau_k' = Z_{ij}'(\tau_k^-) \end{aligned} \tag{4·54}$$

In the above equation, $\tilde{\mu}_{ij}(\tau_k^+) = \mu_{ij}$ because $X_i(\tau_k^+) > 0$. Also, $\mu_{ij} p_{ij}(\tau_k) = \sigma_i(\tau_k)$ and $\tilde{\mu}_{ij}(\tau_k^-) = \frac{\sigma_i(\tau_k)}{p_{ij}(\tau_k)}$ results in $\tilde{\mu}_{ij}(\tau_k^+) = \mu_{ij}$. For $Z_{il}(t)$, $l \neq j$, agent l cannot be connected to target i at τ_k so we have, $Z_{il}'(\tau_k^+) = Z_{il}'(\tau_k^-)$ and $\frac{d}{dt} Z_{il}'(t) = 0$ in $[\tau_k, \tau_{k+1})$. For $Z_{rl}(t)$,$r \neq i$ and $l \neq j$ using the assumption that two events occur at the same time w.p. 0, the dynamics of $Z_{rl}(t)$ are not affected at τ_k, hence we get $\frac{d}{dt} Z_{rl}'(t)$ from (4·49) for i and j replaced by r and l.

3. **Event** ζ_{ij}^0: This event causes a transition from $Z_{ij}(t) > 0$ for $t < \tau_k$ to $Z_{ij}(t) = 0$ for $t \geq \tau_k$. The switching function is $g_k(\Theta, \mathbf{X}) = Z_{ij}(t)$ so $\frac{\partial g_k}{\partial Z_{ij}} = 1$. From (3·19):

$$\begin{aligned} \tau_k{}' &= -\Big(\frac{\partial g_k}{\partial Z_{ij}} f_k(\tau_k^-)\Big)^{-1}\Big(g_k' + \frac{\partial g_k}{\partial Z_{ij}} Z_{ij}'(\tau_k^-)\Big) \\ &= -\frac{Z_{ij}'(\tau_k^-)}{\tilde{\mu}_{ij}(\tau_k^-)p_{ij}(\tau_k^-) - \beta_{ij}p_{B_j}(\tau_k^-)} = \frac{Z_{ij}'(\tau_k^-)}{\beta_{ij}p_{B_j}(\tau_k)} \end{aligned} \tag{4·55}$$

Since $Z_{ij}(t)$ is being emptied at τ_k, by the assumption that agents can not be in range with the base and targets at the same time, we have $p_{ij}(\tau_k) = 0$. Then from (3·16):

$$\begin{aligned} Z_{ij}'(\tau_k^+) &= Z_{ij}'(\tau_k^-) + \Big[-\beta_{ij}p_{B_j}(\tau_k) - 0\Big]\tau_k{}' \\ &= Z_{ij}'(\tau_k^-) - \Big[\beta_{ij}p_{B_j}(\tau_k)\Big]\frac{Z_{ij}'(\tau_k^-)}{\beta_{ij}p_{B_j}(\tau_k)} = 0 \end{aligned} \tag{4·56}$$

Since $\dot{Z}_{ij}(t) = 0$ in $[\tau_k, \tau_{k+1})$:

$$\frac{d}{dt}Z_{ij}'(t) = \frac{\partial \dot{Z}_{ij}(t)}{\partial Z_{ij}(t)}Z_{ij}'(t) + \dot{Z}_{ij}'(t) = 0 \tag{4·57}$$

For $Z_{rl}(t)$, $r \neq i$ or $l \neq j$, the dynamics in (4·7) are not affected at τ_k, hence:

$$Z_{rl}'(\tau_k^+) = Z_{rl}'(\tau_k^-) \tag{4·58}$$

if $Z_{rl}(\tau_k) > 0$, the value for $\frac{d}{dt}Z_{rl}'(t)$ is calculated by (4·49) with r and l replacing i and j respectively. If $Z_{rl}(\tau_k) = 0$ then $\frac{d}{dt}Z_{rl}'(t) = 0$.

For $Y_i(t)$ we have $\beta_i(\tau_k^+) = 0$ since the agent has emptied its queue, hence:

$$\begin{aligned} Y_i'(\tau_k^+) &= Y_i'(\tau_k^-) + \Big[\dot{Y}_i(\tau_k^-) - \dot{Y}_i(\tau_k^+)\Big]\tau_k' \\ &= Y_i'(\tau_k^-) + [\beta_{ij}p_{B_j}(\tau_k) - 0]\frac{Z_{ij}'(\tau_k^-)}{\beta_{ij}p_{B_j}(\tau_k)} \\ &= Y_i'(\tau_k^-) + Z_{ij}'(\tau_k^-) \end{aligned} \tag{4·59}$$

In $[\tau_k, \tau_{k+1})$ we can get $\frac{d}{dt}Y_i'(t) = 0$. For $Y_r(t)$, $r \neq i$ the dynamics of $Y_r(t)$ in (4·9) are not affected by the event at τ_k hence, $Y_r'(\tau_k^+)$ and $\frac{d}{dt}Y_r'(t)$ in $[\tau_k, \tau_{k+1})$ are calculated from (4·45) and (4·46) respectively. The dynamics of $X_r(t)$, $r = 1, \ldots, M$ is are not affected at τ_k since the event at τ_k is happening at the base. We have $X_r'(\tau_k^+) = X_r'(\tau_k^-)$. If $X_r(\tau_k) > 0$ then we have $\frac{d}{dt}X_r'(t)$ from (4·44) and if $X_r(\tau_k) = 0$ then $\frac{d}{dt}X_r'(t) = 0$ in $[\tau_k, \tau_{k+1})$.

4. **Event** δ_{ij}^+: This event causes a transition from $d_{ij}^+(t) = 0$ for $t \leq \tau_k$ to $d_{ij}^+(t) > 0$ for to $t > \tau_k$. It is the moment that agent j leaves target i's range. The switching function is $g_k(\Theta, \mathbf{X}) = d_{ij}(t) - r_{ij}$, from (3·19):

$$\tau_k{}' = -\frac{\partial d_{ij}}{\partial s_j} s_j'(t) \Big(\frac{\partial d_{ij}}{\partial s_j} \dot{s}_j(\tau_k) \Big)^{-1} \tag{4·60}$$

If agent j was connected to target i at τ_k then by leaving the target, it is possible that another agent l which is within range with target i connects to that target. This means $\dot{X}_i(\tau_k^+) = \sigma_i(\tau_k) - \mu_{il} p_{il}(\tau_k)$ and $\dot{X}_i(\tau_k^-) = \sigma_i(\tau_k) - \mu_{ij} p_{ij}(\tau_k)$, with $p_{ij}(\tau_k) = 0$, from (3·16) we have

$$X_i'(\tau_k^+) = X_i'(\tau_k^-) - \mu_{il} p_{il}(\tau_k) \tau_k' \tag{4·61}$$

If $X_i(\tau_k) > 0$, $\frac{d}{dt}X_i'(t)$ in $[\tau_k, \tau_{k+1})$ is as in (4·44) with r replaced by i and if $X_i(\tau_k) = 0$ then $\frac{d}{dt}X_i'(t) = 0$. On the other hand, if agent j was not connected to target i at τ_k, we know that some $l \neq j$ is already connected to target i. This means agent j leaving target i cannot affect the dynamics of $X_i(t)$ so we have $X_i'(\tau_k^+) = X_i'(\tau_k^-)$ and $\frac{d}{dt}X_i'(t)$ is calculated from (4·44) with r replaced by i.

For $X_r(t)$, $r \neq i$ the dynamics in (4·5) are not affected by the event at τ_k hence, we get $X_r'(\tau_k^+)$ from (4·43). If $X_r(\tau_k) > 0$ the time derivative $\frac{d}{dt}X_r'(t)$ in $[\tau_k, \tau_{k+1})$ can be calculated from (4·44) and if $X_r(\tau_k) = 0$ then $\frac{d}{dt}X_r'(t) = 0$.

For $Y_r(t)$, $r = 1, \ldots, , M$, the dynamics in (4·9) are not also affected by the event at

τ_k hence, we get $Y_r(\tau_k^+)$ from (4·45) and in $[\tau_k, \tau_{k+1})$ the $\frac{d}{dt}Y_r'(t)$ is calculated from (4·46).

For $Z_{ij}(t)$, the dynamics in (4·7) are not affect at τ_k, regardless of the fact that agent j is connected to target i or not. We have $\dot{Z}_{ij}(\tau_k^-) = \tilde{\mu}_{ij}(\tau_k)p_{ij}(\tau_k)$ with $p_{ij}(\tau_k) = 0$ and $\dot{Z}_{ij}(\tau_k^+) = 0$, hence from (3·16):

$$\begin{aligned} Z_{ij}'(\tau_k^+) &= Z_{ij}'(\tau_k^-) + \left[\dot{Z}_{ij}(\tau_k^-) - \dot{Z}_{ij}(\tau_k^+)\right]\tau_k' \\ &= Z_{ij}'(\tau_k^-) + \tilde{\mu}_{ij}(\tau_k)p_{ij}(\tau_k)\tau_k' = Z_{ij}'(\tau_k^-) \end{aligned} \tag{4·62}$$

and in $[\tau_k, \tau_{k+1})$, we have $\frac{d}{dt}Z_{ij}'(t) = 0$ using (4·49) knowing $p_{ij}(\tau_k) = p_{B_j}(\tau_k) = 0$. For $Z_{rl}(t)$, $r \neq i$ or $l \neq j$, the dynamics of $Z_{rl}(t)$ are not affected at τ_k hence (4·58) holds and in $[\tau_k, \tau_{k+1})$ again we can use (4·49) with i and j replaced by r and l.

5. **Event** δ_{ij}^0: This event causes a transition from $d_{ij}^+(t) > 0$ for $t < \tau_k$ to $d_{ij}^+(t) = 0$ for to $t \geq \tau_k$. The event is the moment that agent j enters target i's range. The switching function is $g_k(\Theta, \mathbf{X}) = d_{ij}(t) - r_{ij}$. From (3·19) we can get ${\tau_k}'$ from (4·60). If no other agent is already connected to target i, agent j connects to it. Otherwise, if another agent is already connected to target i, no connection is established. For $X_i(t)$, the dynamics in (4·5) are not affected in both cases, hence, (4·53) holds. If $X_i(t) > 0$ in $[\tau_k, \tau_{k+1})$ we calculate $\frac{d}{dt}X_i'(t)$ using (4·44) with l being the appropriate connected agent to target i. If $X_i(\tau_k^-) = 0$, $\frac{d}{dt}X_i'(t) = 0$. For $X_r(t)$, $r \neq i$ the dynamics in (4·5) are not affected by the event at τ_k. Hence, we get $X_r'(\tau_k^+)$ from (4·43). If $X_r(\tau_k) > 0$ we calculate $\frac{d}{dt}X_r'(t)$ from (4·44) with i replaced by r and if $X_r(\tau_k) = 0$ then $\frac{d}{dt}X_r'(t) = 0$.

For $Y_r(t)$, $r = 1, \ldots, M$ again the dynamics in (4·9) are not affected at τ_k so both (4·45) and (4·46) hold.

For $Z_{ij}(t)$, with agent j being connected or not to target i at τ_k the dynamics of

$Z_{ij}(t)$ are unaffected at τ_k, hence (4·58) holds for i and j and in $[\tau_k, \tau_{k+1})$ the $\frac{d}{dt}Z'_{ij}(t)$ is calculated through (4·49). For $Z_{rl}(t)$, $r \neq i$ or $l \neq j$ the dynamics are unaffected (4·58) holds again. In $[\tau_k, \tau_{k+1})$, $\frac{d}{dt}Z'_{rl}(t)$ is given through (4·49) with i and j replaced by r and l.

6. **Event** Δ_j^+: This event causes a transition from $d_{Bj}^+(t) = 0$ for $t \leq \tau_k$ to $d_{Bj}^+(t) \geq 0$ for $t > \tau_k$. The switching function is $g_k(\Theta, \mathbf{X}) = d_{B_j}(t) - r_{B_j}$.

$$\tau_k{}' = -\frac{\partial d_{B_j}}{\partial s_j} s'_j(\tau_k) \Big(\frac{\partial d_{B_j}}{\partial s_j} \dot{s}_j(\tau_k) \Big)^{-1} \tag{4·63}$$

Similar to the previous event, the dynamics of $X_i(t)$ are unaffected at τ_k hence, we have $X'_i(\tau_k^+)$ calculated from (4·53). If $X_i(t) > 0$ in $[\tau_k, \tau_{k+1})$ we calculate $\frac{d}{dt}X'_i(t)$ through (4·44) and if $X_i(\tau_k^-) = 0$, $\frac{d}{dt}X'_i(t) = 0$.

For $Y_r(t)$, $r = 1, \ldots, , M$, the dynamics of $Y_r(t)$ in (4·9) are not affected at τ_k, hence, we get $Y_r(\tau_k^+)$ from (4·45) and in $[\tau_k, \tau_{k+1})$, $\frac{d}{dt}Y'_r(t)$ is calculated from (4·46).

For $Z_{ij}(t)$, Using the fact that agent j can only be connected to one target or the base, we have $\dot{Z}_{ij}(\tau_k^-) = \beta_{ij}(\tau_k) p_{B_j}(\tau_k)$ with $p_{B_j}(\tau_k) = 0$ and $\dot{Z}_{ij}(\tau_k^+) = 0$, hence (4·58) holds with i and j replacing r and l. In $[\tau_k, \tau_{k+1})$ from (3·18):

$$\begin{aligned} \frac{d}{dt}Z'_{ij}(t) &= \frac{\partial \dot{Z}_{ij}(t)}{\partial Z_{ij}(t)} Z'_{ij}(t) + \dot{Z}'_{ij}(t) \\ &= \dot{Z}'_{ij}(t) = -\beta_{ij} p'_{B_j}(t) \end{aligned} \tag{4·64}$$

As for $Z_{rl}(t)$, $r \neq i$ or $l \neq j$ the dynamics are unaffected so (4·58) holds. In $[\tau_k, \tau_{k+1})$ we can calculate $\frac{d}{dt}Z'_{rl}(t)$ through (4·49) with j replacing l.

7. **Event** Δ_j^0: This event causes a transition from $d_{Bj}^+(t) > 0$ for $t < \tau_k$ to $d_{Bj}^+(t) = 0$ for $t \geq \tau_k$. The switching function is $g_k(\Theta, \mathbf{X}) = d_{B_j}(t) - r_{B_j}$. Using (3·19) we can get $\tau_k{}'$ from (4·63). Similar with the previous event we have $X'_i(\tau_k^+)$

from (4·53). If $X_i(t) > 0$ we can get $\frac{d}{dt}X_i'(t)$ from (4·44) and if $X_i(\tau_k^-) = 0$ then $\frac{d}{dt}X_i'(t) = 0$.

For $Y_r(t)$, $r = 1, \ldots, , M$, we again follow the previous event analysis so (4·45) and (4·46) hold.

For $Z_{ij}(t)$, the analysis is similar to event Δ_j^+ so we can calculate $Z_{ij}'(\tau_k^+)$ and $\frac{d}{dt}Z_{ij}'(t)$ in $[\tau_k, \tau_{k+1})$ from (4·54) and (4·49) respectively. Also for $Z_{rl}(t)$, $r \neq i$ or $l \neq j$, (4·58) holds with same reasoning as previous event. In $[\tau_k, \tau_{k+1})$ we calculate $\frac{d}{dt}Z_{rl}'(t)$ from (4·49).

In summary, what we have discussed for all events reduces to a simple set of state derivative dynamics as shown next.

Proposition 4.1. *After an event occurrence at $t = \tau_k$, the state derivatives $X_i'(\tau_k^+)$, $Y_i'(\tau_k^+)$, $Z_{ij}'(\tau_k^+)$, with respect to the controllable parameter Θ satisfy the following:*

$$X_i'(\tau_k^+) = \begin{cases} 0 & \text{if } e(\tau_k) = \xi_i^0 \\ X_i'(\tau_k^-) - \mu_{il} p_{il}(\tau_k)\tau_k^{'} & \text{if } e(\tau_k) = \delta_{ij}^+ \\ X_i'(\tau_k^-) & \text{otherwise} \end{cases} \tag{4·65}$$

where $l \neq j$ with $p_{il}(\tau_k) > 0$ if such l exists and $\tau_k^{'} = \frac{\partial d_{ij}(s_j)}{\partial s_j} s_j' \left(\frac{\partial d_{ij}(s_j)}{\partial s_j} \dot{s}_j(\tau_k) \right)^{-1}$.

$$Y_i'(\tau_k^+) = \begin{cases} Y_i'(\tau_k^-) + Z_{ij}'(\tau_k^-) & \text{if } e(\tau_k) = \zeta_{ij}^0 \\ Y_i'(\tau_k^-) & \text{otherwise} \end{cases} \tag{4·66}$$

$$Z_{ij}'(\tau_k^+) = \begin{cases} 0 & \text{if } e(\tau_k) = \zeta_{ij}^0 \\ Z_{ij}'(\tau_k^-) + X_i'(\tau_k^-) & \text{if } e(\tau_k) = \xi_i^0 \\ Z_{ij}'(\tau_k^-) & \text{otherwise} \end{cases} \tag{4·67}$$

where $e(\tau_k) = \xi_i^0$ occurs when j is connected to target i.

Proof. See (4·42), (4·53), (4·61), (4·59), (4·45), (4·54), (4·56), (4·48). □

This result shows that only three of the events in E can actually cause discontinuous changes to the state derivatives. Further, note that $X_i'(t)$ is reset to zero after a ξ_i^0 event. Moreover, when such an event occurs, note that $Z_{ij}'(t)$ is coupled to $X_i'(t)$. Similarly for $Z_{ij}'(t)$ and $Y_i'(t)$ when event ζ_{ij}^0 occurs, showing that perturbations in Θ can only propagate to an adjacent queue when that queue is emptied.

Proposition 4.2. *The state derivatives $X_i'(\tau_{k+1}^-)$, $Y_i'(\tau_{k+1}^-)$ with respect to the controllable parameter Θ satisfy the following after an event occurrence at $t = \tau_k$:*

$$X_i'(\tau_{k+1}^-) = \begin{cases} 0 & \text{if } e(\tau_k) = \xi_i^0 \\ X_i'(\tau_k^+) - \int_{\tau_k}^{\tau_{k+1}} \mu_{ij} p_{ij}'(u)du & \text{otherwise} \end{cases} \tag{4·68}$$

$$Y_i'(\tau_{k+1}^-) = Y_i'(\tau_k^+) + \int_{\tau_k}^{\tau_{k+1}} \beta_i'(u)du \tag{4·69}$$

where j is such that $p_{ij}(t) > 0$, $t \in [\tau_k, \tau_{k+1})$.

Proof. See (4·41), (4·44) and (4·46). □

Proposition 4.3. *The state derivatives $Z_{ij}'(\tau_{k+1}^+)$ with respect to the controllable parameter Θ satisfy the following after an event occurrence at $t = \tau_k$:*
i- *If j is connected to target i,*

$$Z_{ij}'(\tau_{k+1}^-) = \begin{cases} Z_{ij}'(\tau_k^+) & \text{if } e(\tau_k) = \xi_i^0,\ \zeta_{ij}^0 \text{ or } \delta_{ij}^+ \\ Z_{ij}'(\tau_k^+) + \int_{\tau_k}^{\tau_{k+1}} \mu_{ij} p_{ij}'(u)du & \text{otherwise} \end{cases} \tag{4·70}$$

ii- *If j is connected to B with $Z_{ij}(\tau_k) > 0$,*

$$Z_{ij}'(\tau_{k+1}^-) = Z_{ij}'(\tau_k^+) - \int_{\tau_k}^{\tau_{k+1}} \beta_{ij} p_{Bj}'(u)du \tag{4·71}$$

iii- *Otherwise, $Z_{ij}'(\tau_{k+1}^-) = Z_{ij}'(\tau_k^+)$.*

Proof. See (4·49), (4·50), (4·57) and (4·64). □

Corollary 4.1. *The state derivatives $X_i'(t)$, $Z_{ij}'(t)$, $Y_i'(t)$ with respect to the controllable parameter Θ are independent of the random data arrival processes $\{\sigma_i(t)\}$, $i = 1, \ldots, M$.*

Proof. Follows directly from the three Propositions. □

There are a few important consequences of these results. First, as the Corollary asserts, one can apply IPA regardless of the characteristics of the random processes $\{\sigma_i(t)\}$. This robustness property does not mean that these processes do not affect the values of the $X_i'(t)$, $Z_{ij}'(t)$, $Y_i'(t)$; this happens through the values of the event times τ_k, $k = 1, 2, \ldots$, which are observable and enter the computation of these derivatives as seen above. Second, the IPA estimation process is event-driven: $X_i'(\tau_k^+)$, $Y_i'(\tau_k^+)$, $Z_{ij}'(\tau_k^+)$ are evaluated at event times and then used as initial conditions for the evaluations of $X_i'(\tau_{k+1}^-)$, $Y_i'(\tau_{k+1}^-)$, $Z_{ij}'(\tau_{k+1}^-)$ along with the integrals appearing in Propositions 2,3 which can also be evaluated at $t = \tau_{k+1}$. Consequently, this approach is scalable in the number of events in the system as the number of agents and targets increases. Third, despite the elaborate derivations in the Appendix, the actual implementation reflected by the three Propositions is simple. Finally, returning to (4·39), note that the integrals involving $\nabla\mathcal{L}_1(\Theta, t)$, $\nabla\mathcal{L}_2(\Theta, t)$ are directly obtained from $X_i'(t)$, $Y_i'(t)$, the integral involving $\nabla\mathcal{L}_3(\Theta, t)$ is obtained from straightforward differentiation of (4·15), and the final term is obtained from $Z_{ij}'(T)$.

Objective Function Optimization

This is carried out using (4·38) with an appropriate step size sequence.

Elliptical Trajectories: Elliptical trajectories are described by their center coordinates, minor and major axes and orientation. Agent j's position $s_j(t) = [s_j^x(t), s_j^y(t)]$ follows the general parametric equation of the ellipse:

$$\begin{aligned} s_j^x(t) &= A_j + a_j \cos\rho_j(t) \cos\phi_j - b_j \sin\rho_j(t) \sin\phi_j \\ s_j^y(t) &= B_j + a_j \cos\rho_j(t) \sin\phi_j + b_j \sin\rho_j(t) \cos\phi_j \end{aligned} \tag{4·72}$$

Here, $\Theta_j = [A_j, B_j, a_j, b_j, \phi_j]$ where A_j, B_j are the coordinates of the center, a_j and b_j are the major and minor axis respectively while $\phi_j \in [0, \pi)$ is the ellipse orientation which is defined as the angle between the x axis and the major axis of the ellipse. The time dependent parameter $\rho_j(t)$ is the eccentric anomaly of the ellipse. Since the agent is moving with constant speed of 1 on this trajectory from (4·32), we have $\dot{s}_j^x(t)^2 + \dot{s}_j^y(t)^2 = 1$ which gives

$$\dot{\rho}_j(t) = \left[\begin{array}{c} \Big(a \sin\rho_j(t)\cos\phi_j + b_j \cos\rho_j(t)\sin\phi_j\Big)^2 \\ \\ +\Big(a \sin\rho_j(t)\sin\phi_j - b_j \cos\rho_j(t)\cos\phi_j\Big)^2 \end{array} \right]^{-\frac{1}{2}} \tag{4·73}$$

In the data harvesting problem, trajectories that do not pass through the base are inadmissible since there is no delivery of data. Therefore, we add a constraint to force the ellipse to pass through $w_{_B} = [w_{_B}^x, w_{_B}^y]$ where:

$$\begin{aligned} w_{_B}^x =& A_j + a_j \cos\rho_j(t)\cos\phi_j - b_j \sin\rho_j(t)\sin\phi_j \\ w_{_B}^y =& B_j + a_j \cos\rho_j(t)\sin\phi_j + b_j \sin\rho_j(t)\cos\phi_j \end{aligned} \tag{4·74}$$

Using the fact that $\sin^2\rho(t) + \cos^2\rho(t) = 1$ we define a quadratic constraint term added to $J(\Theta, T; \mathbf{X}(\Theta, 0))$ with a sufficiently large multiplier. This can ensure the optimal path passes through the base location. We define $\mathcal{C}_j(\Theta_j)$ which appears in (4·76):

$$\mathcal{C}_j(\Theta_j) = (1 - f_j^1 \cos^2\phi_j - f_j^2 \sin^2\phi_j - f_j^3 \sin 2\phi_j)^2 \tag{4·75}$$

where $f_j^1 = (\frac{w_{_B}^x - A_j}{a_j})^2 + (\frac{w_{_B}^y - B_j}{b_j})^2$, $f_j^2 = (\frac{w_{_B}^x - A_j}{b_j})^2 + (\frac{w_{_B}^y - B_j}{a_j})^2$, $f_j^3 = \frac{(b_j^2 - a_j^2)(w_{_B}^x - A_j)(w_{_B}^y - B_j)}{a_j^2 b_j^2}$.

Multiple visits to the base may be needed during the mission time $[0, T]$. We can capture this by allowing an agent trajectory to consist of a sequence of admissible ellipses. For each agent, we define $\mathcal{E}_j$ as the number of ellipses in its trajectory. The parameter vector Θ_j^κ with $\kappa = 1, \ldots, \mathcal{E}_j$, defines the κ^{th} ellipse in agent j's trajectory

and $\mathcal{T}_j^\kappa$ is the time that agent j completes ellipse κ. Therefore, the location of each agent is described through κ during $[\mathcal{T}_j^{\kappa-1}, \mathcal{T}_j^\kappa]$ where $\mathcal{T}_j^0 = 0$. Since we cannot optimize over all possible $\mathcal{E}_j$ for all agents, an iterative process needs to be performed in order to find the optimal number of segments in each agent's trajectory. At each step, we fix $\mathcal{E}_j$ and find the optimal trajectory with that many segments. The process is stopped once the optimal trajectory with $\mathcal{E}_j$ segments is no better than the optimal one with $\mathcal{E}_j - 1$ segments (obviously, this is not a globally optimal solution). We can now formulate the parametric optimization problem $\mathbf{P4.3_e}$ where $\Theta_j = [\Theta_j^1, \ldots, \Theta_j^{\mathcal{E}_j}]$ and $\Theta = [\Theta_1, \ldots, \Theta_N]$:

$$\begin{aligned}\min_{\Theta \in F_\Theta} J_e = &\frac{1}{T}\int_0^T \Big[qJ_1(\Theta,t) - (1-q)J_2(\Theta,t) + J_3(\Theta,t) + J_4(\Theta,t)\Big]dt \\ &+ M_C \sum_{j=1}^{N} \mathcal{C}_j(\Theta_j) + J_f(\Theta, T)\end{aligned} \tag{4·76}$$

where M_C is a large multiplier. The evaluation of $\nabla\mathcal{C}_j$ is straightforward and does not depend on any event. (Details are shown in Appendix A.2.1).

Fourier Series Trajectories: The elliptical trajectories are limited in shape and may not be able to cover many targets in a mission space. Thus, we next parameterize the trajectories using a Fourier series representation of closed curves (Zahn and Roskies, 1972). Using a Fourier series function for f and g in (4·35), agent j's trajectory can be described as follows with base frequencies f_j^x and f_j^y:

$$\begin{aligned} s_j^x(t) = &\ a_{0,j} + \sum_{n=1}^{\Gamma_j^x} a_{n,j}\sin(2\pi n f_j^x \rho_j(t) + \phi_{n,j}^x) \\ s_j^y(t) = &\ b_{0,j} + \sum_{n=1}^{\Gamma_j^y} b_{n,j}\sin(2\pi n f_j^y \rho_j(t) + \phi_{n,j}^y) \end{aligned} \tag{4·77}$$

The parameter $\rho(t) \in [0, 2\pi]$, similar to elliptical trajectories, represents the position of the agent along the trajectory. In this case, forcing a Fourier series curve to pass

through the base is easier. For simplicity, we assume a trajectory to start at the base and set $s_j^x(0) = w_B^x$, $s_j^y(0) = w_B^y$. Assuming $\rho(0) = 0$, with no loss of generality, we can calculate the zero frequency terms by means of the remaining parameters:

$$a_{0,j} = w_B^x - \sum_{n=1}^{\Gamma_j^x} a_{n,j} \sin(\phi_{n,j}^x), b_{0,j} = w_B^y - \sum_{n=1}^{\Gamma_j^y} b_{n,j} \sin(\phi_{n,j}^y) \tag{4·78}$$

The parameter vector for agent j is

$$\Theta_j = [f_j^x, a_{0,j}, \ldots, a_{\Gamma_j^x}, b_{0,j}, \ldots, b_{\Gamma_j^y}, \phi_{1,j}, \ldots, \phi_{\Gamma_j^x}, \xi_{1,j}, \ldots, \xi_{\Gamma_j^y}]$$

and $\Theta = [\Theta_1, \ldots, \Theta_N]$. Note that the shape of the curve is fully represented by the ratio f_j^x/f_j^y so one of these can be kept constant. For the Fourier trajectories, the fact that $\mathbf{u}_j^* = 1$ allows us to calculate $\dot{\rho}_j(t)$ as follows:

$$\dot{\rho}_j(t) = \frac{1}{2\pi} \left[\begin{array}{c} \left(f_j^x \sum_{n=1}^{\Gamma_j^x} a_{n,j} n \cos(2\pi f_j^x \rho_j(t) + \phi_{n,j}^x) \right)^2 \\ + \left(f_j^y \sum_{n=1}^{\Gamma_j^x} b_{n,j} n \cos(2\pi f_j^y \rho_j(t) + \phi_{n,j}^y) \right)^2 \end{array} \right]^{-1/2} \tag{4·79}$$

Problem $\mathbf{P4.3_f}$ is the same as **P4.3** and there are no additional constraints in this case:

$$\min_{\Theta \in F_\Theta} J_f = \frac{1}{T} \int_0^T \Big[q J_1(\Theta, t) - (1-q) J_2(\Theta, t) + J_3(\Theta, t) + J_4(\Theta, t) \Big] dt + J_f(T) \tag{4·80}$$

4.4 Simulation Results

In this section numerical results are presented to illustrate our approach. The mission space S is considered to be $[0, 10] \times [0, 10]$ in all the presented cases. The first case to consider is a small mission to see the TPBVP results and the fact that it is not scalable to bigger problems.

In Case I we consider a two target - two agent setting. We assume deterministic arrival process with $\sigma_i = 0.5$ for all i. For (4·3) and (4·4) we have used $p(w, v) = \max(0, 1 - \frac{d(w,v)}{r})$ where r is the corresponding value of r_{ij} or $r_{_{Bj}}$. We have $\mu_{ij} = 100$ and $\beta_{ij} = 500$ for all i and j. Other parameters used are $q = 0.5$, $r_{ij} = r_{_{Bj}} = 0.5$ and $T = 20$. The trajectories comparison from TPBVP, Elliptical and Fourier parametric solution is shown in Figures 4·4, 4·5, 4·6. In each figure, the trajectories are shown in top while the actual objective function decline and convergence in the middle graph. The lower graph shows the total amount of data at targets at any time (In blue) and the total amount of data at the base (In green).

In the TPBVP results, the main limitation is the size of the time step that can be considered since the number of control values grows with the number of time steps. To bring this into prospective, for this sample problem with T=20 we considered 300 time steps meaning 300 value for the heading of each agent needs to be calculated which brings the total number of controls to 600. In contrast for the same problem the total number of controls for the elliptical trajectories are 10 parameters and for the fourier trajectories is 28. This explains why the TPBVP can not be a viable solution for larger value of T.

In Table 4.2, the actual values for J^*, J_1^*, J_2^* are shown for the three different trajectories of Figure 4·4.,4·5,4·6. Note that the objective is to minimize J by minimizing J_1 and maximizing J_2.

Next in Case II we consider 9 targets and 2 agents. The base is located at the

Table 4.2: Results Comparison for Case I

Method	J^*	J_1^*	J_2^*
TPBVP	0.272	0.098	0.038
Elliptical	0.255	0.092	0.095
Fourier	0.202	0.089	0.095

Table 4.3: Results Comparison for Case II

Method	J^*	J_1^*	J_2^*
Elliptical	0.19	0.090	0.124
Fourier	0.18	0.069	0.117

center of the mission space. We have $\sigma_i(t) = 0.5$, $\mu_{ij} = 50$ and $\beta_{ij} = 500$ for all i and j. Other parameters used are $q = 0.5$, $r_{ij} = 0.55$, $r_{_{Bj}} = 0.65$ and $T = 50$. In Figure 4·7 the solution with two ellipses in each agent's trajectory is shown. As can be seen the trajectory correctly finds all the target locations and empties the target queues periodically. In Figure 4·8 the fourier trajectories is shown. The two graphs on the bottom of the figures show the objective function value and instantaneous total content at targets and base. The simulation results for Case II are in Table 4.3.

Table 4.4: Results Comparison for Case III

Method	J^*	J_1^*	J_2^*
Elliptical	0.35	0.12	0.09
Fourier	0.23	0.09	0.1
Fourier (Stochastic Arrival)	0.23	0.13	0.13

The third case has 12 targets that are uniformly distributed in the mission space. Here we try to examine the robustness of our approach with respect to the arrival

rate process at targets. We use the same parameters as in case II and solve the problem for deterministic $\sigma_i(t) = 0.5$ using the elliptical trajectories and fourier trajectories. The same mission is then simulated assuming that $\sigma_i(t)$ is a stochastic process with piecewise linear arrival rate. The average arrival rate is kept at 0.5. The results in Figure 4·9,4·10,4·11 and Table 4.4 shows that using the Fourier parametric trajectories, almost same performance can be achieved by the optimization algorithm in the stochastic settings.

4.5 Comparison with a Graph Based Algorithm

The final parametric trajectories can provide us with a sequence of targets visit, same as in a tour selection algorithm that uses the underlying graph topology of the mission space to come up with sequences.

We have compared the results of our approach with a graph topology algorithm called Path Splitter Heuristic (PSH) developed in (Moazzez-Estanjini and Paschalidis, 2012). This algorithm start with the best hamiltonian sequence and then uses a heuristic method to divide the hamiltonian tour into several sub-tours that go through a few targets and then go back to the base. The algorithm then provides a sequence of these sub-tours for each agent. We compare the sequences from Case I and Case II in both elliptical and fourier trajectories and results are shown in Table 4.5 and 4.5. For a fair comparison we adopt each sequence and apply it with the same system dynamic in our model. Meaning, the agent's can pick up the data once within range of the targets. This however is not the basics modeling assumptions used in PSH where agents pick up all the data at the target simultaneously once at the target location. We compare the sum total of data at targets and the base for T=200. Longer time is used for these comparison to get close to an infinite time results. These sequences are shown in Figure 4·12, 4·13 and 4·14. Each color represents one agent's trajectory.

We can see from these comparisons that in the graph-based approach targets are completely divided between agents. This generates a spatial partitioning, giving each agent full responsibility of a set of targets. However, in the trajectory planning results, in most cases, we see more of a temporal partitioning where agents can visit same targets but in different time of the mission. This can allow for more robustness with respect to agent's failure or other changes in an agent's condition.

Table 4.5: Results Comparison with PSH for Case II

Method	J_1^*	J_2^*
PSH Sequence	0.023	0.22
Elliptical Sequence	0.027	0.21
Fourier Sequence	0.024	0.21

Table 4.6: Results Comparison with PSH for Case III

Method	J_1^*	J_2^*
PSH Sequence	0.0257	0.21
Elliptical Sequence	0.0304	0.199
Fourier Sequence	0.0212	0.21

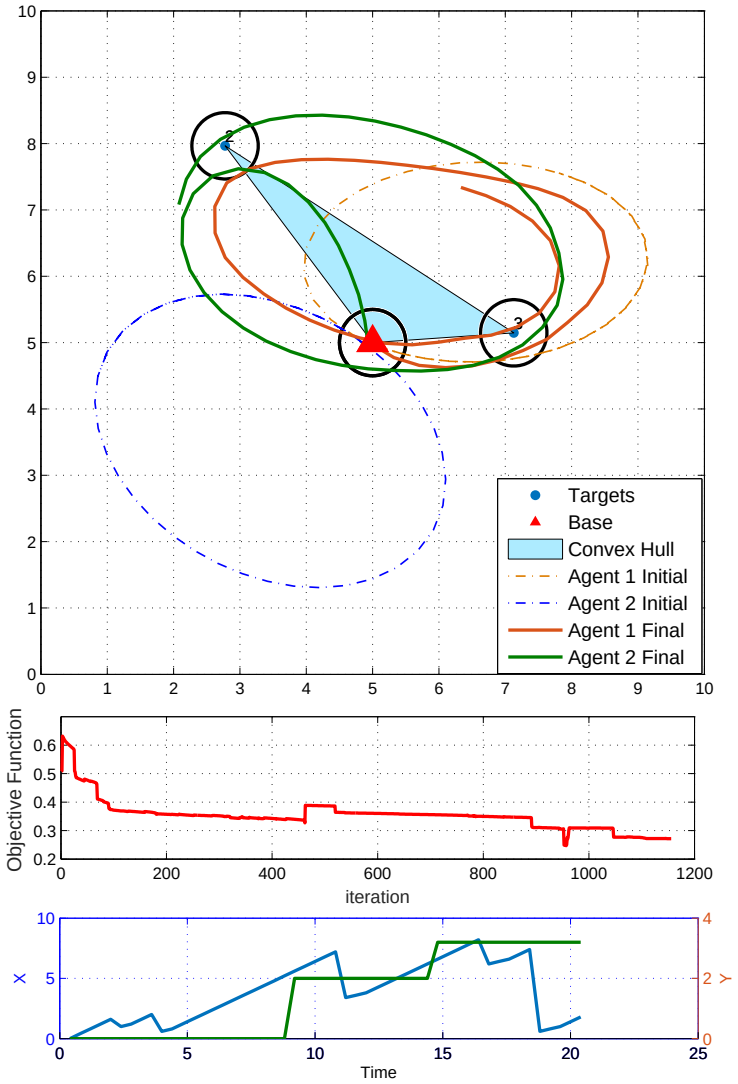


Figure 4·4: TPBVP Trajectories for case I

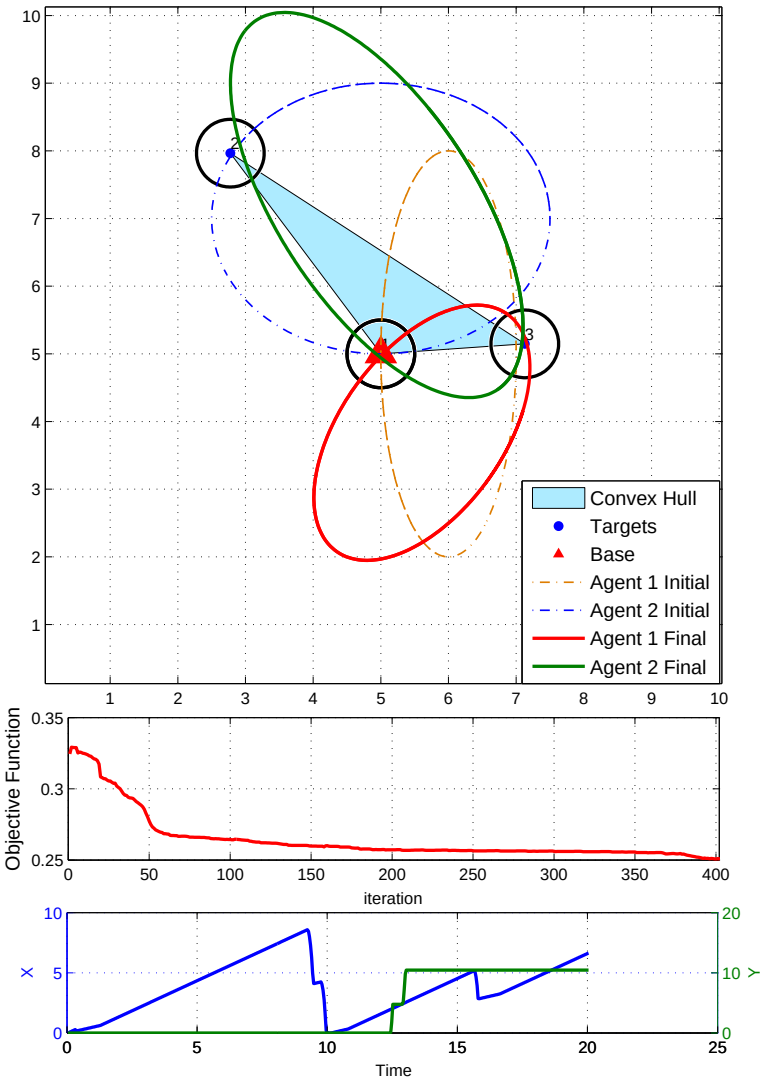


Figure 4·5: Elliptical Trajectories for case I

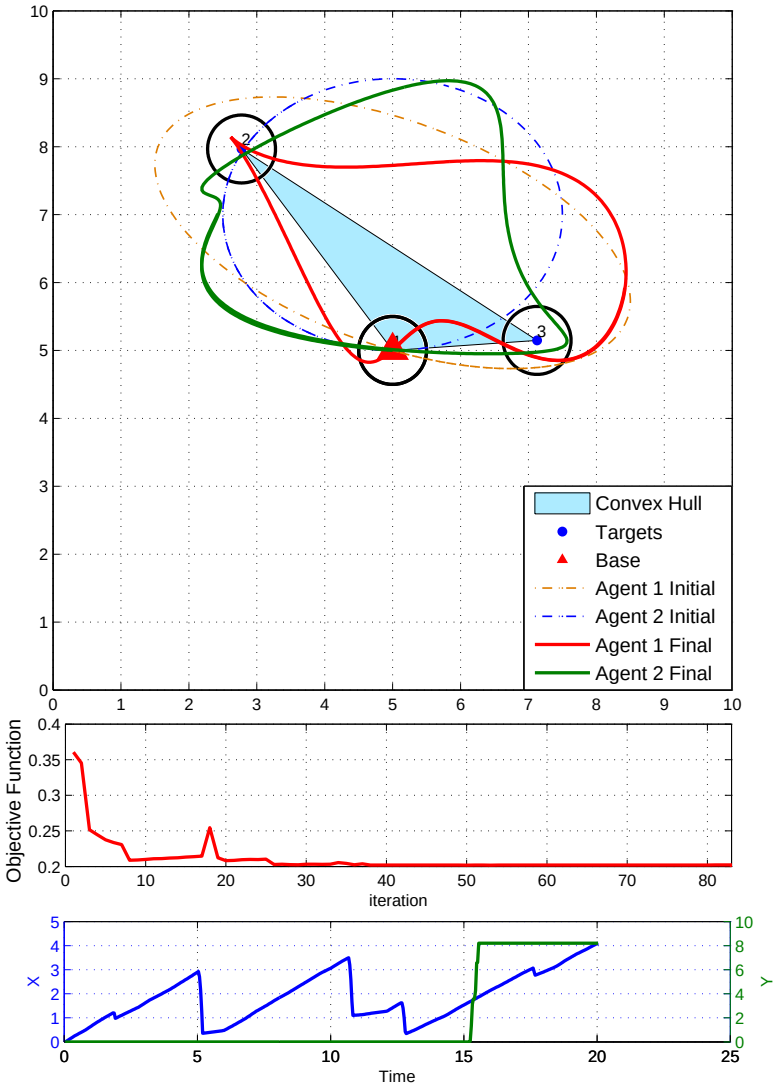


Figure 4·6: Fourier Trajectories for case I

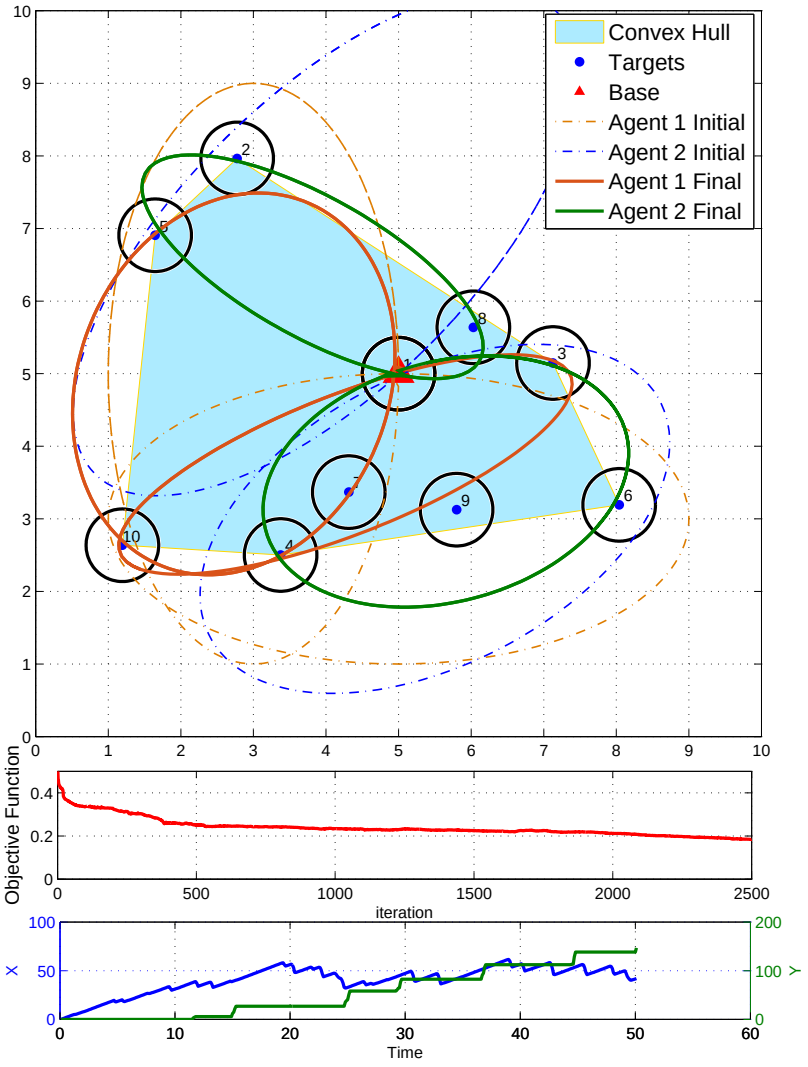


Figure 4·7: Elliptical Trajectories for Case II

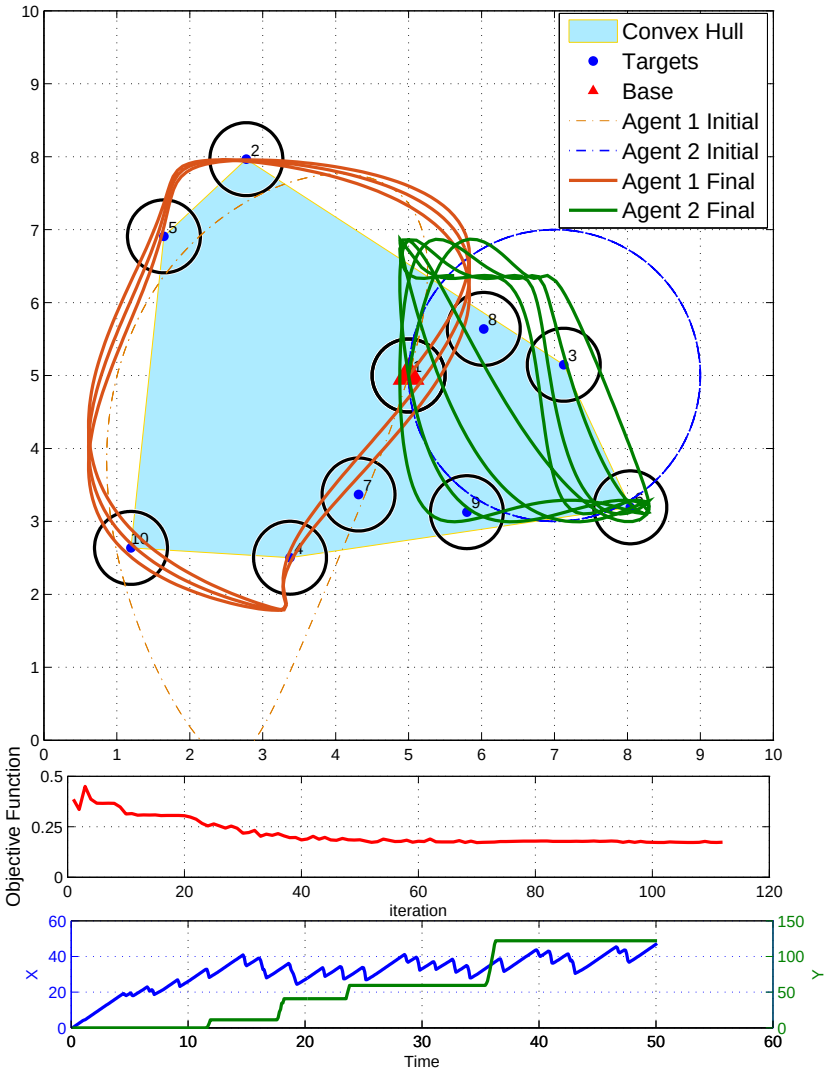


Figure 4·8: Fourier Trajectories for Case II

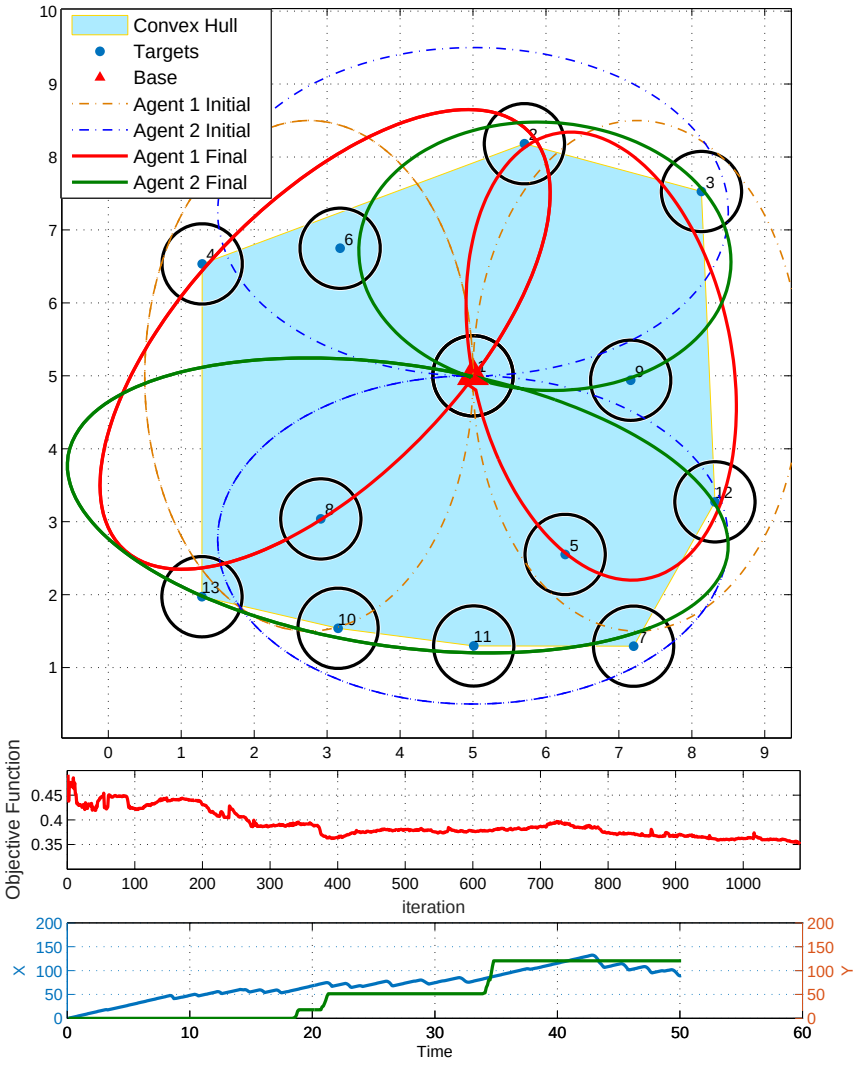


Figure 4·9: Elliptical Trajectories for case III

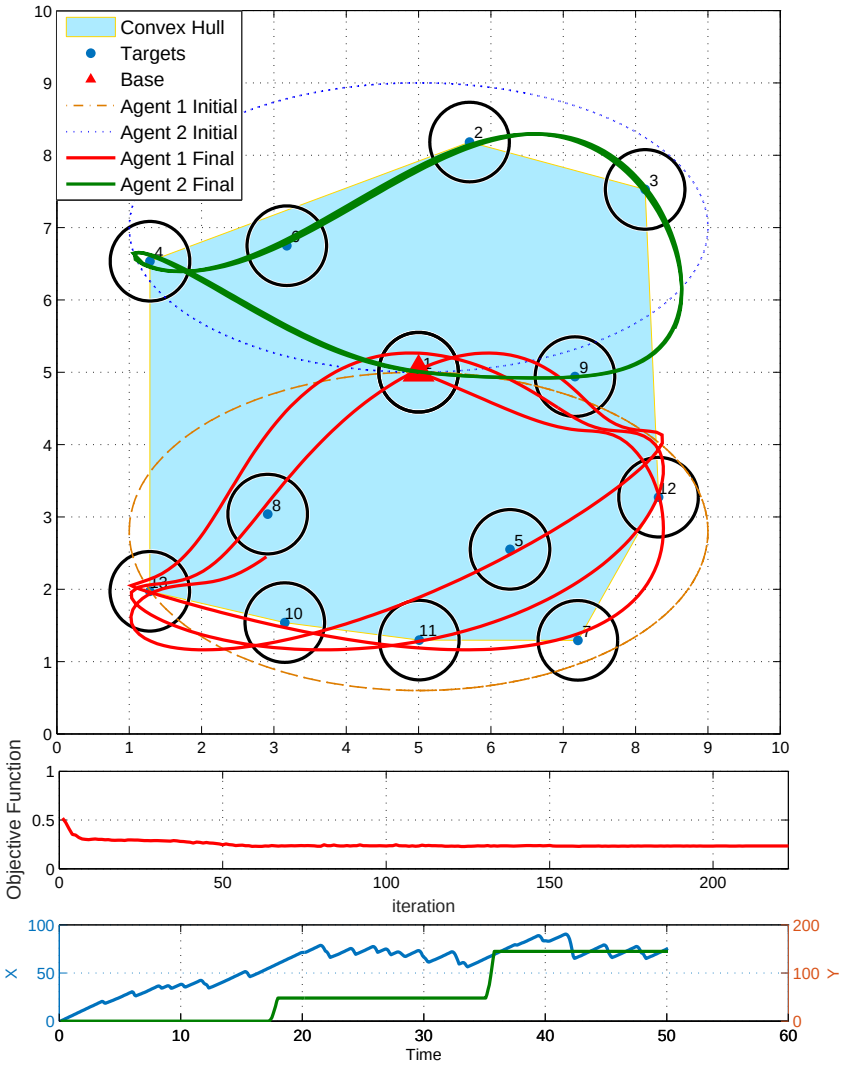


Figure 4·10: Fourier Trajectories for case III

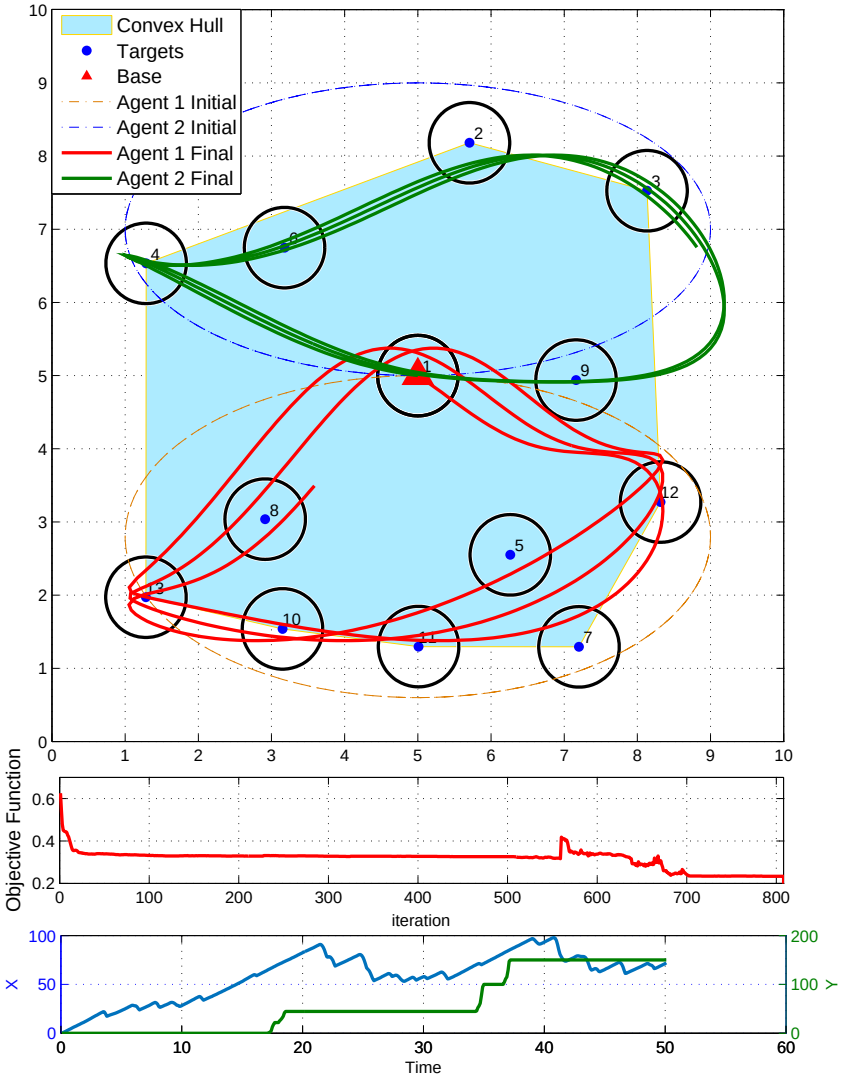


Figure 4·11: Fourier Trajectories for case III with Stochastic Arrival

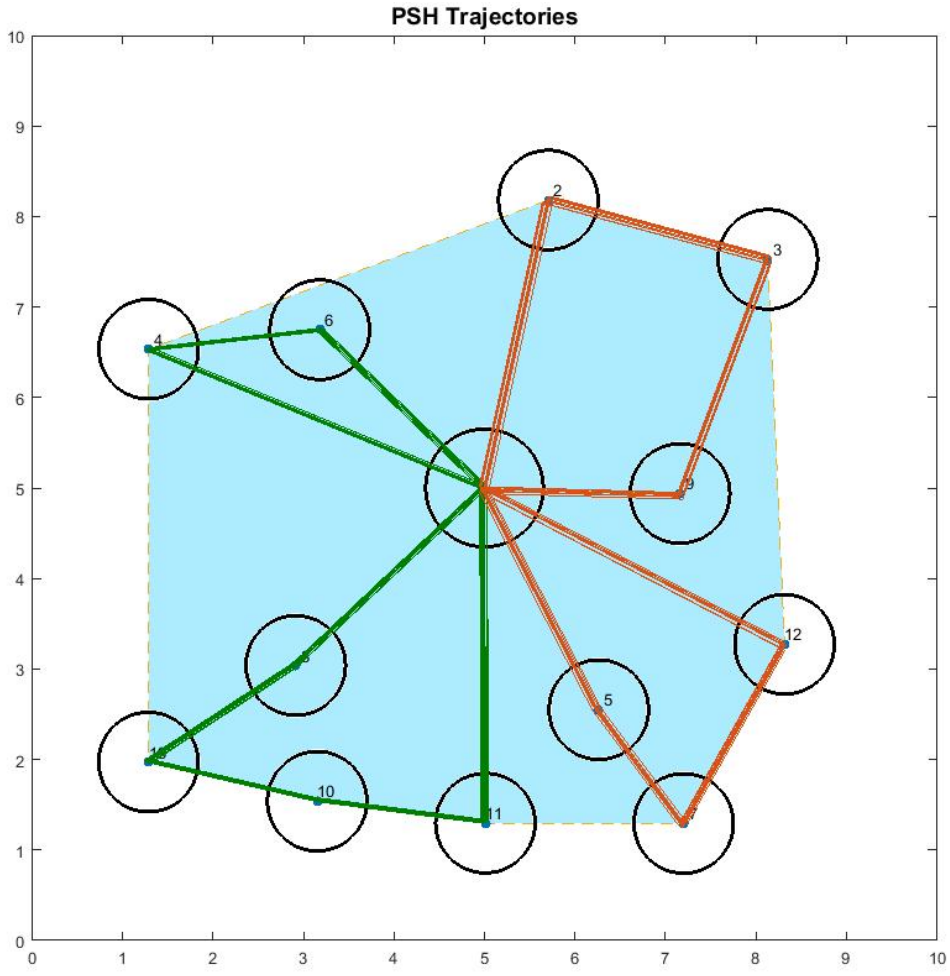


Figure 4·12: PSH Sequences for case III

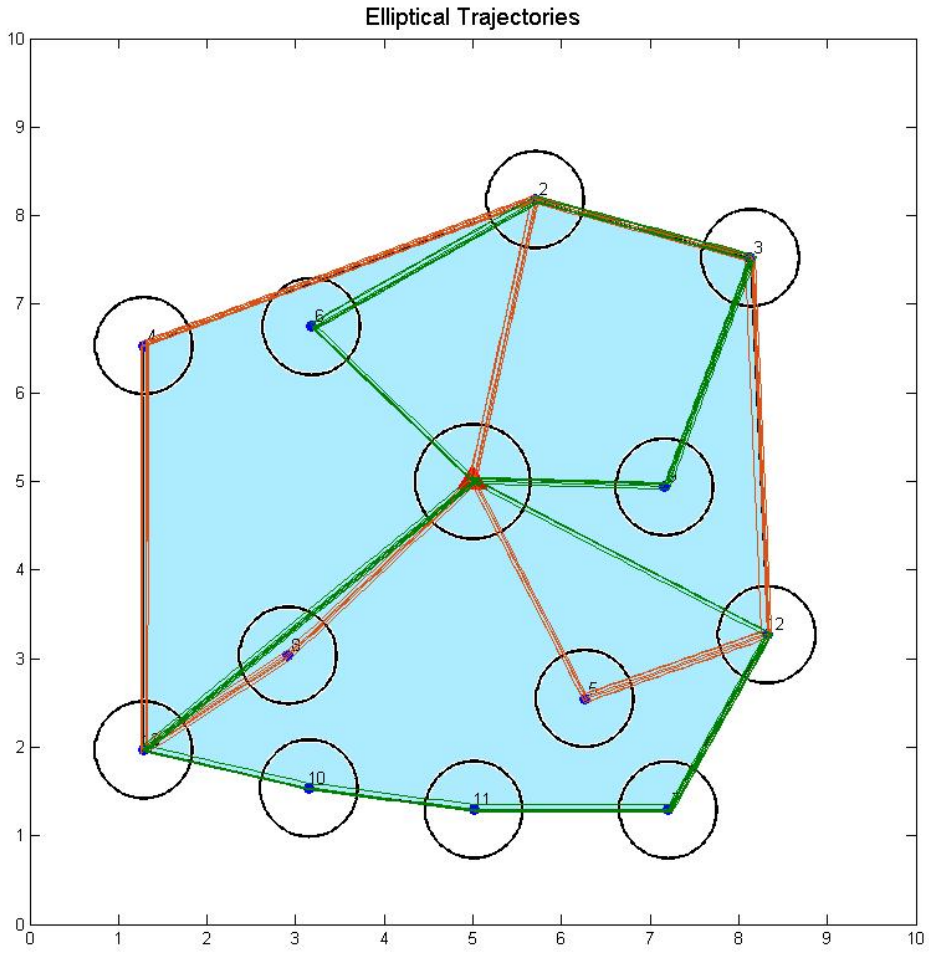


Figure 4·13: Elliptical Sequences for case III

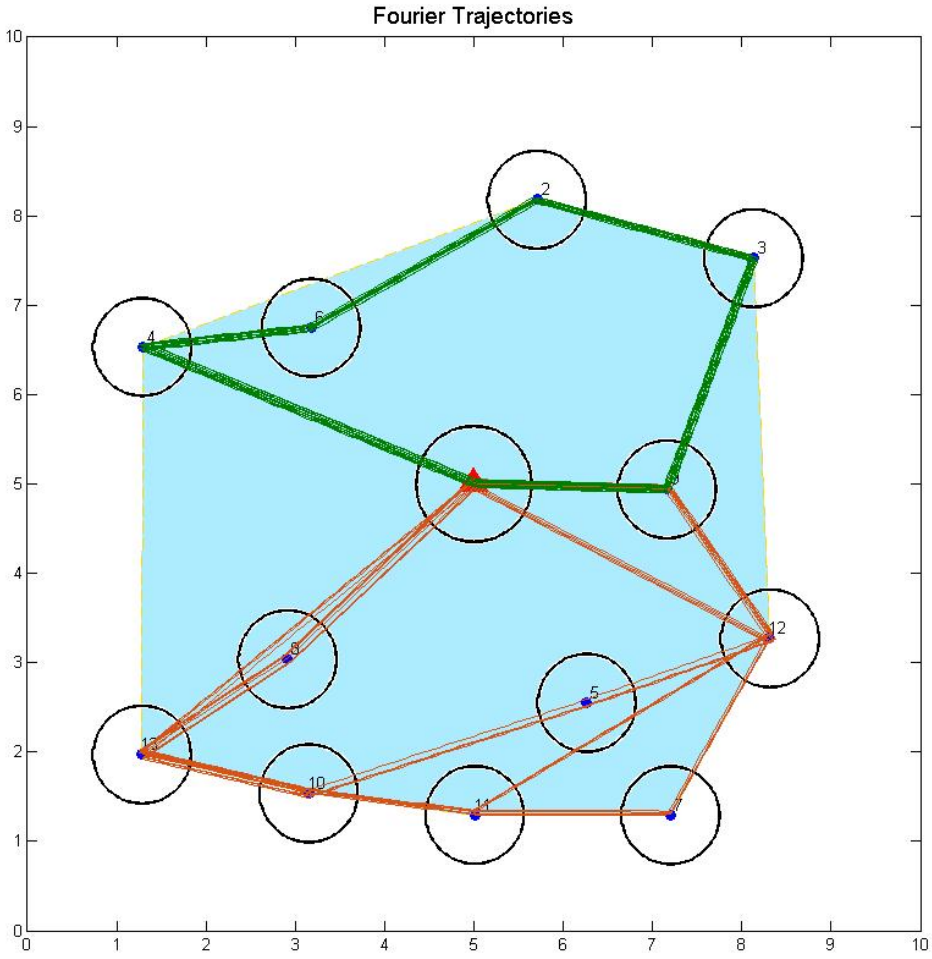


Figure 4·14: Fourier Sequences for case III

Chapter 5

Conclusions and Future Directions

In this work we focused on applying several event-driven control methodologies to different classes of multi-agent systems. At first we proposed an event-driven cooperative receding horizon controller for the maximum reward collection problem. The CRH controller, finds the (local)optimal heading for the agents in the problem by defining an finite time optimization problem for a planning horizon and then the control is actuated for the duration of action horizon which is smaller than or equal to the planning horizon.

The length of planning horizon and action horizon are determined based on the topology and observed events of the system. The uncertainty in the environment could be unknown location of targets, appearing and disappearing of the targets at random times, threats and obstacles and agent's failure. Some of these uncertainties have been studied and the CRH controller is able to respond to them by re-solving the optimal control problem at the time an event happens and new information become available.

Building on the fundamental structure of the previously introduced Cooperative Receding Horizon Controller in (Li and Cassandras, 2006b) we have tackled several shortcomings of the original work. These shortcomings are studied and improved with a totally new definition for the problem's objective function.

We have introduced the idea of active target set, which describes the possible best visits for each agent. Using the active target set definition, we have shown that

the feasible optimal heading for each agent at each time step of the problem lies in a finite set of headings. This improves the speed of the algorithm and tend to result in a lot less hedging and instability of the agents.

We also have introduced the notion of travel cost factor, which generates a new distance metric between agents and targets. This metric allows us to determine the reward-to-go in the objective function on a projected shortest path for the agent with respect to the travel cost factor. This provides us with a more realistic value for the estimated reward at the end of the mission, in contrast to the unattainable loose upper bound that was used in the previous work.

The performance comparison of the new CRH controller with previous work, shows significant improvement in the final collected rewards for randomly generated cases. We also see a great improvement in eliminating the instabilities in the agent's trajectories.

In the second part of this work we have focused on application of event-driven trajectory planning in a few classes of multi-agent systems. At first we introduce a general framework of multi-agent systems moving toward classes with discrete point of interests as targets. We model this system as a general stochastic hybrid system while pointing at fundamental issues that exist when trying to apply event-driven optimization methods to solve the problem. We introduce and address the issue of event excitation in this general class of problems. A new metric is introduced that captures the agent and target interaction in a continuous way while the point of interests make a finite discrete set in the mission space.

The general problem is modeled as a finite time optimal control that needs a TP-BVP numerical solver. Due to computational complexity and benefits of parametric trajectories, we introduce a few set of parametrization for the proposed trajectories. The parametric optimization problem is then solved using gradient base methods

while applying the Infinitesimal Perturbation Analysis techniques in estimating the performance measure derivatives with respect to the trajectory parameters.

The methodology is applied to two class of problems. The first on is a data collection model which reduces to TSP like problems in the simplest case. The second is the data harvesting problem in which data is collected from targets and delivered to the base. The goal is to minimize the target's data content and maximize the base data content. We have compared the results from our trajectory optimization with a graph-based tour selection approach called path splitter heuristic (PSH). We have shown that the new methodology can achieve comparable results and in some cases does better. It can be seen that the type of cooperation in methods like PSH is more of a spatial partitioning while the trajectories tend to achieve the cooperation in a temporal aspect and visit targets in different time schedule rather than dividing targets between each other. On of the benefits of this is that in case of agent's failure the impact that the system gets might be more tolerable. However, we normally converge to a local optimal trajectory which depends on the initial trajectory and this can affect the end results if the optimization is not initialized with a good choice.

Although the proposed methodology is focused on applying the event-driven optimization approach to the problem of data harvesting or data collection, it has to be noted that the new metric that has been introduced allows us to generalize the methodology to other applications as well. The new metric introduces a potential field or density map over the whole mission space, this can be looked at as a probability distribution of potential targets in problems where the exact location of targets are unknown. This probability density can be used a prior info which can be improved while the agents move around the mission space and gather more information. In a further step this density can be dynamically changing if the targets are moving or their location changes with time, this obviously needs some prior information of the

target's moving path. This provides us with a tool to apply the trajectory optimization in a much broader range of problems where targets are tracking points of interest. Also for instanced of the general persistent monitoring or coverage problems, where there are only finite points that are of interest, the new metric can provide a tool to apply event-driven optimization methods as well.

5.1 Future Directions

5.1.1 Extensions for CRH controller:

The receding horizon controller can be applied to other multi-agent systems and specially as the very first step it can be applied to the data harvesting problem which we have addressed from a different angel. The collection delay might be modeled as a negative reward while the delivery delay is modeled as a dynamic reward at the base.

The CRH controller can also incorporate environment's constraints and obstacles. We may model the problem in graph topology and in special case the grid topology, with non-Euclidean distance metrics incorporating the same ideas of active target set and travel cost factor.

We also showed that the final result of the CRH controller is very sensitive to the quality of the reward-to-go estimation. We improved this estimation compared to the upper bound in the previous work. However, it might still be further improved using heuristic methods and over the shelf minimum tour estimation techniques to find faster and better estimations.

5.1.2 Extensions for Trajectory Optimization:

The trajectory optimization methodology can also be applied to problems like tracking and surveillance where even the targets locations are not fully known. The density function that creates the potential field can easily be replaced by a probability

distribution which carries information about possible target locations. An on-line application of such controller can be used to "learn" the actual location of the targets through iterative realization of the system.

The mission space can also be extended to more interesting setting, such as no-pass zones and obstacles. Accounting for obstacles might be handled by introducing targets with negative reward. These can model an obstacle in terms of an area that an agent has no interest in and is repelled from visiting.

5.1.3 Agent's Model Extensions:

In both methods we can extend the models to handle agent's mobility limitation. The first direction might be to consider non-holonomic motion constraints in which the agent is described as a double integrator with limited angular velocity. We can also consider agents that have a limited capacity to carry rewards or data. This is a realistic model for problems such as disaster planning or evacuations along with general sensor networks where the buffer size is limited. The other direction to consider is teams of non-uniform agents which have different dynamics and capabilities. This can allow us to analyse fleets of agents in transportation or smart city problems.

5.1.4 Extension to Decentralized Control Methods

Both methods can be extended to decentralized controllers by considering the right communication protocols between agents. The CRH controller has been extended to a distributed version in its original format and the new methodology can easily allow for each agent to calculate its own optimal heading if it knows the targets location within its own partition. The amount of data and the type of communication have to be discussed precisely for each specific problem. In the case of trajectory optimization, also extending the methodology to a distributed scheme with limited information is possible.

Appendices

Appendix A

Mathematical Proofs

A.1 Chapter 2

A.1.1 Proof of Lemma 2.1

Proof. Before proving the lemma we prove $\forall \mathbf{x} \in \mathcal{F}_j(t_k, H_k)$

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) \leq \eta_l(\mathbf{x}, t_k + H_k) \tag{A·1}$$

From the definition of $\eta_i(x,t)$ in (2·18) and $\mathcal{C}_{l,j}(t_k, H_k)$ in (2·22) we have:

$$d(\mathcal{C}_{l,j}(t_k, H_k), \mathbf{y}_l) \leq d(\mathbf{x}, \mathbf{y}_l),\ \forall \mathbf{x} \in \mathcal{F}_j(t_k, H_k) \tag{A·2}$$

dividing both sides of (A·2) by $\lambda_l d_l^{-1}$ and adding $\zeta_l(t_k + H_k)$ we get the results in (A·1). Now we prove the actual lemma 2.1 starting from the forward statement:
$''\Longrightarrow''$: To prove the forward statement, we prove the if target l is an active target then (2·23) is true. We use contradiction assuming there exist a target r such that

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) > \eta_r(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) \tag{A·3}$$

Using (A·1) we get $\forall \mathbf{x} \in \mathcal{F}_j(t_k, H_k)$

$$\eta_r(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) < \eta_l(\mathbf{x}, t_k + H_k) \tag{A·4}$$

This means $\nexists\, \mathbf{x} \in \mathcal{F}_j(t_k, H_k)$ s.t. $l = \text{argmin}_i\, \eta_i(\mathbf{x}, t_k + H_k)$. Therefore target l can not be an active target which contradicts the assumption, hence (2·23) is true.
$''\Longleftarrow''$: To prove the reverse statement, assuming $\forall i \in \mathcal{T}_k$,

$$\eta_l(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) < \eta_i(\mathcal{C}_{l,j}(t_k, H_k), t_k + H_k) \tag{A·5}$$

By the definition of active targets, we know that target l is an active target of agent j at time t_k. □

A.1.2 Proof of Lemma 2.2

Proof. Active target set creates a partition on the set $\mathcal{F}_j(t_k, H_k)$ where each partition is an arc in a Euclidean mission space. For an active target $l \in S_j(t_k, H_k)$ we call this partition $\mathcal{F}_j^l(t_k, H_k) \subset \mathcal{F}_j(t_k, H_k)$. For each partition $\mathcal{F}_j^l(t_k, H_k)$ we prove that the heading $\mathbf{v}^* = v(\mathcal{C}_{l,1}(t_k, H_k))$ satisfies, $\forall \mathbf{x} \in \mathcal{F}_j^l(t_k, H_k)$

$$J_{\mathbf{I}}(\mathbf{v}^*, t_k, H_k) + J_{\mathbf{A}}(\mathbf{v}^*, t_k, H_k) > J_{\mathbf{I}}(v(\mathbf{x}), t_k, H_k) + J_{\mathbf{A}}(v(\mathbf{x}), t_k, H_k) \tag{A·6}$$

We consider two different cases:

1: $\mathbf{y}_l \in \mathcal{F}_1(t_k, H_k)$, This means $d(\mathbf{y}_l, \mathbf{x}_1(t)) = H_k$. Also following the definition $\mathcal{C}_{l,1}(t_k, H_k) = \mathbf{y}_l$ in (2·22). This guarantees that $\forall r \in \mathcal{T}_k$:

$$q_r(\mathcal{C}_{r,1}(t_k + H_k)) = \begin{cases} 1 & \text{if } r = l \\ 0 & \text{,o.w.} \end{cases} \tag{A·7}$$

so assuming $\tilde{\tau}_r(\mathbf{v}^*, t_k, H_k)) = \tilde{\tau}_r^*$ we have

$$\begin{aligned} J(\mathbf{v}^*, t_k, H_k) =& J_{\mathbf{I}}(\mathbf{v}^*, t_k, H_k) + J_{\mathbf{A}}(\mathbf{v}^*, t_k, H_k) \\ =& \lambda_l \phi_l(t_k + H_k) + \sum_{r=1}^{M_{k+1}} \lambda_r \phi_r(\tilde{\tau}_r^*) q_l(\mathbf{x}_1(\tilde{\tau}_r^*)) \end{aligned}$$

Here $M_{k+1} = |\mathcal{T}_{k+1}|$ and $\mathcal{T}_{k+1} = \mathcal{T}_k - \{l\}$ since target l will be already collected at time $t_k + H_k$. $\tilde{\tau}_r^*$ will be determined based on a tour $\boldsymbol{\theta}$ that starts at point $\mathbf{y}_l$. Now let's calculate the objective function for any other heading $v(\mathbf{x})$ where $\mathbf{x} \in \mathcal{F}_1^l(t_k, H_k)$. Let's call $\tilde{\tau}_r(v(\mathbf{x}), t_k, H_k)) = \tilde{\tau}_r$

$$\begin{aligned} J(v(\mathbf{x}), t_k, H_k) =& J_{\mathbf{I}}(v(\mathbf{x}), t_k, H_k) + J_{\mathbf{A}}(v(\mathbf{x}), t_k, H_k) \\ =& \sum_{r=1}^{M_k} \lambda_r \phi_r(t_k + H_k) q_r(\mathbf{x}(t_k + H_k)) + J_{\mathbf{A}}(v(\mathbf{x}), t_k, H_k) \\ =& 0 + \sum_{r=1}^{M'_{k+1}} \lambda_r \phi_r(\tilde{\tau}_r) q_l(\mathbf{x}_1(\tilde{\tau}_r)) \end{aligned}$$

Because $\mathbf{x} \neq \mathcal{C}_{l,1}(t_k, H_k)$ so $q_r(\mathbf{x}) = 0$ for all $r \in \mathcal{T}_k$, therefore the immediate reward is equal to 0. The aggregated tour is determined over the set $\mathcal{T}'_{k+1} = \mathcal{T}_k$ and it starts at point $\mathbf{x} \in \mathcal{F}_1^l(t_k, H_k)$. By the definition the target with the least travel cost from point $\mathbf{x}$ is the active target l and this will be the first target in the tour. The rest of the tour consists of targets in $\mathcal{T}_{k+1} - l$ starting at $\mathbf{y}_l$. Let's call this tour $\boldsymbol{\theta}'$. Since in both tours $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ the starting point and the set of available targets are the same, the order of the targets will be identical and we have $\boldsymbol{\theta}' = [l \quad \boldsymbol{\theta}]$. The collection times in the $\boldsymbol{\theta}$ can be calculated as:

$$\tilde{\tau}^*_{\boldsymbol{\theta}(n),1} = t_k + H_k + \sum_{i=1}^{n-1} d(\mathbf{y}_{\boldsymbol{\theta}(i)}, \mathbf{y}_{\boldsymbol{\theta}(i+1)}) \tag{A·8}$$

In the second tour the collection time for target $\boldsymbol{\theta}'(1) = l$ is: $\tilde{\tau}_{\boldsymbol{\theta}'(1),1} = t_k + H_k + d(\mathbf{x}, \mathbf{y}_l)$. For the rest of the targets with $1 < n \leq M'_{k+1}$

$$\tilde{\tau}_{\boldsymbol{\theta}'(n),1} = t_k + H_k + d(\mathbf{x}, \mathbf{y}_l) + \sum_{i=1}^{n-1} d(\mathbf{y}_{\boldsymbol{\theta}'(i)}, \mathbf{y}_{\boldsymbol{\theta}'(i+1)}) \tag{A·9}$$

$\forall 1 < n \leq M_{k+1}$ we have $\boldsymbol{\theta}'(n+1) = \boldsymbol{\theta}(n)$, now we can say: $\tilde{\tau}_{\boldsymbol{\theta}'(n+1),1} > \tilde{\tau}_{\boldsymbol{\theta}(n),1}$. We assumed that for all $i \in \mathcal{T}$, $\phi_i(t)$ is a non-increasing discount function, therefore $\phi_{\boldsymbol{\theta}'(n+1)}(\tilde{\tau}_{\boldsymbol{\theta}'(n+1),1}) \leq \phi_{\boldsymbol{\theta}(n)}(\tilde{\tau}_{\boldsymbol{\theta}(n),1})$, so we have

$$\lambda_l \phi_l(t_k + H_k + d(\mathbf{x}, \mathbf{y}_l)) + \sum_{n=2}^{M'_{k+1}} \lambda_{\boldsymbol{\theta}'(n)} \phi_{\boldsymbol{\theta}'(n)}(\tilde{\tau}_{\boldsymbol{\theta}'(n),1}) \leq \lambda_l \phi_l(t_k + H_k) + \sum_{n=1}^{M_k+1} \lambda_{\boldsymbol{\theta}(n)} \phi_{\boldsymbol{\theta}(n)}(\tilde{\tau}_{\boldsymbol{\theta}(n),1}) \tag{A·10}$$

The right side is $J(\mathbf{v}^*, t_k, H_k)$ and the left side is $J(v(\mathbf{x}), t_k, H_k)$, so we have proved that $\forall \mathbf{x} \in \mathcal{F}_1^l(t_k, H_k)$ where $\mathbf{x} \neq \mathcal{C}_{l,1}(t_k, H_k)$ we have $J(v(\mathbf{x}), t_k, H_k) \leq J(\mathbf{v}^*, t_k, H_k)$.

2: $\mathbf{y}_l \notin \mathcal{F}_1(t_k, H_k)$, In this case for any point $\mathbf{x} \in \mathcal{F}_j^l(t_k, H_k)$ we have a zero immediate reward. Thus the aggregated reward is to be compared. Using (2·20), for any $\mathbf{x} \in \mathcal{F}_j^l(t_k, H_k)$ we know the aggregation tour $\boldsymbol{\theta}$ for any point $\mathbf{x}$ starts with target l and the rest of it would also be the same. Now similarly let's assume $\boldsymbol{\theta}$ shows the tour for $\mathbf{v}^*$ and $\boldsymbol{\theta}'$ shows the tour for any other point $\mathbf{x}$. The collection times for $\boldsymbol{\theta}$ are:

$$\tilde{\tau}^*_{\boldsymbol{\theta}(n),1} = t_k + H_k + d(\mathbf{y}_l, \mathcal{C}_{l,1}(t_k, H_k)) + \sum_{i=1}^{n-1} d(\mathbf{y}_{\boldsymbol{\theta}(i)}, \mathbf{y}_{\boldsymbol{\theta}(i+1)}) \tag{A·11}$$

and for $\boldsymbol{\theta}'$:

$$\tilde{\tau}^{*}_{\boldsymbol{\theta}(n),1} = t_k + H_k + d(\mathbf{y}_l, \mathbf{x}) + \sum_{i=1}^{n-1} d(\mathbf{y}_{\boldsymbol{\theta}(i)}, \mathbf{y}_{\boldsymbol{\theta}(i+1)}) \tag{A·12}$$

Now by the definition in (2·22), $\mathcal{C}_{l,1}(t_k, H_k))$ is on the shortest path from $\mathbf{x}_j(t_k)$ to $\mathbf{y}_l$ meaning $\tilde{\tau}_{\boldsymbol{\theta}'(n),1} > \tilde{\tau}_{\boldsymbol{\theta}(n),1}$. Again with the reward discount function being non-increasing we have $\phi_{\boldsymbol{\theta}'(n)}(\tilde{\tau}_{\boldsymbol{\theta}'(n),1}) \leq \phi_{\boldsymbol{\theta}(n)}(\tilde{\tau}_{\boldsymbol{\theta}(n),1})$. Which means $J(v(\mathbf{x}), t_k, H_k) \leq J(\mathbf{v}^*, t_k, H_k)$. We proved that for all the active targets the best heading among the headings associated with that target is the direct heading towards that. This means the optimal heading would be one of the direct headings toward an active target. □

A.1.3 Proof of Theorem 2.1

Proof. In the multiple agent mission, calculating the reward-to-go in (2·29) for each agent is exactly like a one agent mission only on its own partition $\mathcal{T}_{k,j}$. Therefore the results follows simply from lemma 2.2. □

A.1.4 Proof of Theorem 2.2

Proof. We assume WLOG $d(\mathbf{x}, \mathbf{y}_1) < d(\mathbf{x}, \mathbf{y}_2)$. Now let's assume we are at time step t_k so we have that $H_k = d(\mathbf{x}, \mathbf{y}_1)$. As shown in fig. 2·12 target 1 will be on agent 1's feasible location for next time step. This implies that target 1 is always an active target (The travel cost of target 1 at time $t_k + H_k$ is equal to 0). Point $\mathcal{C}_{2,1}(t_k, H_k)$ defined in equation (2·22) is shown in fig. 2·12. We will refer to this point as $\mathcal{C}_{2,1}$. We have $d(\mathbf{x}, \mathbf{y}_1) = d(\mathbf{x}, \mathcal{C}_{2,1}) = H_k$. This results in:

$$d(\mathbf{x}, \mathbf{y}_2) = d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_2, \mathcal{C}_{2,1}) \tag{A·13}$$

From lemma 2.1 we know that target 2 is an active target if and only if

$$\eta_2(\mathcal{C}_{2,1}, t_k + H_k) \leq \eta_1(\mathcal{C}_{2,1}, t_k + H_k) \tag{A·14}$$

From (2·18), target 2 is an active target if and only if:

$$\frac{d(\mathcal{C}_{2,1}, \mathbf{y}_2)}{\lambda_2 d_2^{-1}} \leq \frac{d(\mathcal{C}_{2,1}, \mathbf{y}_1)}{\lambda_1 d_1^{-1}} \tag{A·15}$$

Simplifying this we have:

$$\frac{\lambda_1}{d_1} d(\mathcal{C}_{2,1}, \mathbf{y}_2) \leq \frac{\lambda_2}{d_2} d(\mathcal{C}_{2,1}, \mathbf{y}_1) \tag{A·16}$$

Now let's assume target 2 is not an active target, meaning:

$$\frac{\lambda_1}{d_1} d(\mathcal{C}_{2,1}, \mathbf{y}_2) > \frac{\lambda_2}{d_2} d(\mathcal{C}_{2,1}, \mathbf{y}_1) \tag{A·17}$$

Let's start with a trivial inequality:

$$0 > \frac{-\lambda_1}{d_1}[d(\mathcal{C}_{2,1}, \mathbf{y}_2) + d(\mathcal{C}_{2,1}, \mathbf{y}_1)] \tag{A·18}$$

Add $\frac{\lambda_2}{d_2}[d(\mathcal{C}_{2,1}, \mathbf{y}_1)]$ to both sides of the inequality:

$$\frac{\lambda_2}{d_2}[d(\mathcal{C}_{2,1}, \mathbf{y}_1)] > \frac{-\lambda_1}{d_1}[d(\mathcal{C}_{2,1}, \mathbf{y}_2)] + (\frac{\lambda_2}{d_2} - \frac{\lambda_1}{d_1}) d(\mathcal{C}_{2,1}, \mathbf{y}_1)] \tag{A·19}$$

Following (A·17) we replace the left hand side with a greater value:

$$\frac{\lambda_1}{d_1}[d(\mathcal{C}_{2,1}, \mathbf{y}_2)] > \frac{-\lambda_1}{d_1}[d(\mathcal{C}_{2,1}, \mathbf{y}_2)] + (\frac{\lambda_2}{d_2} - \frac{\lambda_1}{d_1}) d(\mathcal{C}_{2,1}, \mathbf{y}_1)] \tag{A·20}$$

Add the positive quantity of $\frac{\lambda_2}{d_2}[d(\mathcal{C}_{2,1}, \mathbf{y}_2)]$ to both sides:

$$\begin{aligned} (\frac{\lambda_1}{d_1} + \frac{\lambda_2}{d_2})[d(\mathcal{C}_{2,1}, \mathbf{y}_2)] &> (\frac{\lambda_2}{d_2} - \frac{\lambda_1}{d_1})[d(\mathcal{C}_{2,1}, \mathbf{y}_2)] + (\frac{\lambda_2}{d_2} - \frac{\lambda_1}{d_1})[d(\mathcal{C}_{2,1}, \mathbf{y}_1)] \\ &= (\frac{\lambda_2}{d_2} - \frac{\lambda_1}{d_1})[d(\mathcal{C}_{2,1}, \mathbf{y}_1) + d(\mathcal{C}_{2,1}, \mathbf{y}_2)] \\ &> (\frac{\lambda_2}{d_2} - \frac{\lambda_1}{d_1}) d(\mathbf{y}_1, \mathbf{y}_2) \quad (\textit{Triangle Inequality}) \end{aligned} \tag{A·21}$$

Rearranging the last inequality and using (A·13), we can get:

$$\begin{aligned} &\frac{\lambda_1}{d_1}[d(\mathbf{x}, \mathbf{y}_2) + d(\mathbf{y}_2, \mathbf{y}_1)] + \frac{\lambda_2}{d_2} d(\mathbf{x}, \mathbf{y}_2) \\ > &\frac{\lambda_1}{d_1} d(\mathbf{x}, \mathbf{y}_1) + \frac{\lambda_2}{d_2}[d(\mathbf{x}, \mathbf{y}_1) + d(\mathbf{y}_2, \mathbf{y}_1)] \end{aligned} \tag{A·22}$$

Result in (A·22) is the same as in (2·42) for path $\mathbf{x}_1(t) \rightarrow \mathbf{y}_1 \rightarrow \mathbf{y}_2$ to be optimal. Since target 1 is the only active target at time t_k, the CRH controller decision at

t_k would be to go toward that target. Hence the controller finds the optimal path correctly. Now let's assume both targets are active targets. Let u_1 and u_2 be the headings that go toward target 1 and 2 respectively. This means $\mathbf{x}_1(t_k + H_k, u_1) = \mathbf{y}_1$ and $\mathbf{x}_1(t_k + H_k, u_2) = \mathcal{C}_{2,1}$, Let's write the objective function of the CRH controller for both of these headings.

$$\begin{aligned} J(u_1, t_k, H_k) &= J_{\mathbf{I}}(u_1, t_k, H_k) + J_{\mathbf{A}}(u_1, t_k, H_k) \\ &= \lambda_1 \phi_1(t_k + H_k) + \lambda_2 \phi_2(t_k + H_k + d(\mathbf{y}_1, \mathbf{y}_2)) \end{aligned} \tag{A·23}$$

$$\begin{aligned} J(u_2, t_k, H_k) &= J_{\mathbf{I}}(u_2, t_k, H_k) + J_{\mathbf{A}}(u_2, t_k, H_k) \\ &= 0 + [\lambda_2 \phi_2(t_k + H_k + d(\mathcal{C}_{2,1}, \mathbf{y}_2)) + \lambda_1 \phi_1(t_k + H_k + d(\mathcal{C}_{2,1}, \mathbf{y}_2) + d(\mathbf{y}_1, \mathbf{y}_2)) \end{aligned} \tag{A·24}$$

Note that in order to calculate the objective function for u_2 we find a tour starting at point $\mathcal{C}_{2,1}$ which goes to the target with minimum collections cost. However, for target 2 to be active at t_k it has to have the smallest travel cost at that point. This results in the $J_{\mathbf{A}}(u_2, t_k, H_k)$ to be the reward of going to target 2 and then target 1. We can see that using the reward of each path from (2·37) and (2·38) we can write:

$$J(u_1, t_k, H_k) = R_{1\to 2} \qquad , \qquad J(u_2, t_k, H_k) = R_{2\to 1} \tag{A·25}$$

This simply means the objective function of the CRH controller is equal to the exact reward of the two possible paths. Hence the CRH controller will pick the correct optimal heading at t_k.

□

A.2 Chapter 4

A.2.1 Elliptical Trajectories

In order to calculate the IPA derivatives we need to have the derivative of agent's state variable with respect to all the parameter vector $\Theta_j = [A_j, B_j, a_j, b_j, \phi_j]$ for all agents j. These derivatives do not depend on the events happening in the system since the trajectories of agents are fixed at each iteration. For now we assume $\mathcal{E}_j = 1$

for all $j = 1, \ldots, N$ hence, we drop the superscript. We have:

$$\frac{\partial s_j^x}{\partial A_j} = 1, \qquad \frac{\partial s_j^x}{\partial B_j} = 0 \tag{A·26}$$

$$\frac{\partial s_j^x}{\partial a_j} = \cos\rho_j(t)\cos\phi_j, \qquad \frac{\partial s_j^x}{\partial b_j} = -\sin\rho_j(t)\sin\phi_j \tag{A·27}$$

$$\frac{\partial s_j^x}{\partial \phi_j} = -a_j\cos\rho_j(t)\sin\phi_j - b_j\sin\rho_j(t)\cos\phi_j \tag{A·28}$$

$$\frac{\partial s_j^y}{\partial A_j} = 0, \qquad \frac{\partial s_j^y}{\partial B_j} = 1 \tag{A·29}$$

$$\frac{\partial s_j^y}{\partial a_j} = \cos\rho_j(t)\sin\phi_j, \qquad \frac{\partial s_j^y}{\partial b_j} = \sin\rho_j(t)\cos\phi_j \tag{A·30}$$

$$\frac{\partial s_j^y}{\partial \phi_j} = a_j\cos\rho_j(t)\cos\phi_j - b_j\sin\rho_j(t)\sin\phi_j \tag{A·31}$$

Also the time derivative of the position state variables are calculated as below:

$$\dot{s}_j^x(t) = -a_j\dot{\rho}_j(t)\sin\rho_j(t)\cos\phi_j + b_j\dot{\rho}_j(t)\cos\rho_j(t)\sin\phi_j \tag{A·32}$$

$$\dot{s}_j^y(t) = -a_j\dot{\rho}_j(t)\sin\rho_j(t)\sin\phi_j + b_j\dot{\rho}_j(t)\cos\rho_j(t)\cos\phi_j \tag{A·33}$$

The gradient of the last term in the J_e in (4·76) needs to be calculated separately. We have for $j \neq l$, $\frac{\partial \mathcal{C}_j}{\partial \Theta_l} = 0$ and for $j = l$:

$$\frac{\partial \mathcal{C}_j}{\partial A_j} = 2\mathcal{C}_j\big(-\cos^2\phi_j\frac{\partial f_j^1}{\partial A_j} - \sin^2\phi_j\frac{\partial f_j^2}{\partial A_j} - \sin 2\phi_j\frac{\partial f_j^3}{\partial A_j}\big) \tag{A·34}$$

$$\frac{\partial \mathcal{C}_j}{\partial B_j} = 2\mathcal{C}_j\big(-\cos^2\phi_j\frac{\partial f_j^1}{\partial B_j} - \sin^2\phi_j\frac{\partial f_j^2}{\partial B_j} - \sin 2\phi_j\frac{\partial f_j^3}{\partial B_j}\big) \tag{A·35}$$

$$\frac{\partial \mathcal{C}_j}{\partial a_j} = 2\mathcal{C}_j\big(-\cos^2\phi_j\frac{\partial f_j^1}{\partial a_j} - \sin^2\phi_j\frac{\partial f_j^2}{\partial a_j} - \sin 2\phi_j\frac{\partial f_j^3}{\partial a_j}\big) \tag{A·36}$$

$$\frac{\partial \mathcal{C}_j}{\partial b_j} = 2\mathcal{C}_j\big(-\cos^2\phi_j\frac{\partial f_j^1}{\partial b_j} - \sin^2\phi_j\frac{\partial f_j^2}{\partial b_j} - \sin 2\phi_j\frac{\partial f_j^3}{\partial b_j}\big) \tag{A·37}$$

$$\frac{\partial \mathcal{C}_j}{\partial \phi_j} = 2\mathcal{C}_j((f_j^1 - f_j^2)\sin 2\phi_j - 2f_j^3 \cos 2\phi_j) \tag{A·38}$$

where

$$\begin{aligned} \frac{\partial f_j^1}{\partial A_j} &= -2\Big(\frac{w_B^x - A_j}{a_j^2}\Big), \quad \frac{\partial f_j^1}{\partial B_j} = -2\Big(\frac{w_B^y - B_j}{b_j^2}\Big) \\ \frac{\partial f_j^1}{\partial a_j} &= -2\Big(\frac{(w_B^x - A_j)^2}{a_j^3}\Big), \quad \frac{\partial f_j^1}{\partial b_j} = -2\Big(\frac{(w_B^y - B_j)^2}{b_j^3}\Big) \end{aligned} \tag{A·39}$$

$$\begin{aligned} \frac{\partial f_j^2}{\partial A_j} &= -2\Big(\frac{w_B^x - A_j}{b_j^2}\Big), \ \frac{\partial f_j^2}{\partial B_j} = -2\Big(\frac{w_B^y - B_j}{a_j^2}\Big) \\ \frac{\partial f_j^2}{\partial a_j} &= -2\Big(\frac{(w_B^y - B_j)^2}{a_j^3}\Big), \ \frac{\partial f_j^2}{\partial b_j} = -2\Big(\frac{(w_B^x - A_j)^2}{b_j^3}\Big) \end{aligned} \tag{A·40}$$

$$\begin{aligned} \frac{\partial f_j^3}{\partial A_j} &= -\Big(\frac{(b_j^2 - a_j^2)(w_B^y - B_j)}{a_j^2 b_j^2}\Big) \\ \frac{\partial f_j^3}{\partial B_j} &= -\Big(\frac{(b_j^2 - a_j^2)(w_B^x - A_j)}{a_j^2 b_j^2}\Big) \end{aligned} \tag{A·41}$$

$$\begin{aligned} \frac{\partial f_j^3}{\partial a_j} &= -2\Big(\frac{(w_B^x - A_j)(w_B^y - B_j)}{a_j^3}\Big) \\ \frac{\partial f_j^3}{\partial b_j} &= 2\Big(\frac{(w_B^x - A_j)(w_B^y - B_j)}{b_j^3}\Big) \end{aligned} \tag{A·42}$$

A.2.2 Fourier Series Trajectories

In the fourier parametric trajectories the agent's state derivative is calculated as the following. The parameter vector is

$$\Theta_j = [f_j^x, a_{0,j}, \ldots, a_{\Gamma_j^x}, b_{0,j}, \ldots, b_{\Gamma_j^y}, \phi_{1,j}, \ldots, \phi_{\Gamma_j^x}, \xi_{1,j}, \ldots, \xi_{\Gamma_j^y}]$$

So we have:

$$\frac{\partial s_j^x}{\partial a_{0,j}} = 1, \qquad \frac{\partial s_j^x}{\partial b_{0,j}} = 0 \tag{A·43}$$

$$\frac{\partial s_j^x}{\partial a_{n,j}} = \sin(2\pi n f_j^x \rho_j(t) + \phi_{n,j}^x), \qquad \frac{\partial s_j^x}{\partial b_{n,j}} = 0 \tag{A·44}$$

$$\frac{\partial s_j^x}{\partial \phi_{n,j}^x} = a_{n,j}\cos(2\pi n f_j^x \rho_j(t) + \phi_{n,j}^x) \qquad \frac{\partial s_j^x}{\partial \phi_{n,j}^y} = 0 \tag{A·45}$$

$$\frac{\partial s_j^x}{\partial f_j^x} = 2\pi\rho_j(t)\sum_{n=1}^{\Gamma_j^x} a_{n,j} n\cos(2\pi n f_j^x \rho_j(t) + \phi_{n,j}^x), \tag{A·46}$$

$$\frac{\partial s_j^y}{\partial b_{0,j}} = 1, \qquad \frac{\partial s_j^y}{\partial a_{0,j}} = 0 \tag{A·47}$$

$$\frac{\partial s_j^y}{\partial b_{n,j}} = \sin(2\pi n f_j^y \rho_j(t) + \phi_{n,j}^y), \qquad \frac{\partial s_j^y}{\partial a_{n,j}} = 0 \tag{A·48}$$

$$\frac{\partial s_j^y}{\partial \phi_{n,j}^y} = b_{n,j}\cos(2\pi n f_j^y \rho_j(t) + \phi_{n,j}^y) \qquad \frac{\partial s_j^y}{\partial \phi_{n,j}^x} = 0 \tag{A·49}$$

$$\frac{\partial s_j^y}{\partial f_j^x} = 0 \tag{A·50}$$

Also the time derivative of the position state variables are calculated as below:

$$\dot{s}_j^x(t) = \dot{\rho}_j(t)\sum_{n=1}^{\Gamma_j^x} 2\pi n f_j^x a_{n,j}\cos(2\pi n f_j^x \rho_j(t) + \phi_{n,j}^x), \tag{A·51}$$

$$\dot{s}_j^y(t) = \dot{\rho}_j(t)\sum_{n=1}^{\Gamma_j^y} 2\pi n f_j^y b_{n,j}\cos(2\pi n f_j^y \rho_j(t) + \phi_{n,j}^y), \tag{A·52}$$

A.2.3 Objective Function Gradient

From (4·39) we have:

$$\begin{aligned}\nabla\mathcal{L}(\Theta, T; \mathbf{X}(\Theta; 0))) =& \frac{1}{T}\Big[\sum_{k=0}^{K}\int_{\tau_k}^{\tau_{k+1}} \Big(q\nabla\mathcal{L}_1(\Theta, t) \\ & - (1-q)\nabla\mathcal{L}_2(\Theta, t) + \nabla\mathcal{L}_3(\Theta, t) + \nabla\mathcal{L}_4(\Theta, t)\Big)dt\Big] \\ & + \nabla\mathcal{L}_f(\Theta, T)\end{aligned} \tag{A·53}$$

We calculate each term separately:

$$\nabla\mathcal{L}_1(\Theta,t) = \frac{1}{M_X}\sum_{i=1}^{M} q_i X_i'(t) \tag{A·54}$$

$$\nabla\mathcal{L}_2(\Theta,t) = \frac{1}{M_Y}\sum_{i=1}^{M} q_i Y_i'(t) \tag{A·55}$$

$$\nabla\mathcal{L}_3(\Theta,t) = \frac{1}{M_I I_j(t)}\left(d_{B_j}^{+\prime}(t)\prod_{i=1}^{M} d_{ij}^{+}(t) + d_{B_j}^{+}(t)\sum_{l=1}^{M} d_{lj}^{+\prime}(t)\prod_{i=1,i\neq l}^{M} d_{ij}^{+}(t)\right) \tag{A·56}$$

$$\begin{aligned}\nabla\mathcal{L}_4(\Theta,t) &= \frac{1}{M_P}\sum_{j=1}^{N}\Big[\int_S \Big(R(w,t)+R_{B_j}(w,t)\Big)P_j'(w,t)dw \\ &\quad + \int_S \Big(R'(w,t)+R_{B_j}'(w,t)\Big)P_j(w,t)dw\Big] \\ &= \frac{1}{M_P}\sum_{j=1}^{N}\Big[\int_S \Big(R(w,t)+R_{B_j}(w,t)\Big)2\langle s_j(t)-w, s_j'(t)\rangle dw \\ &\quad + \int_S \Big(\sum_{i=1}^{M}\frac{q_i X_i'(t)}{d_i^{+}(w)} + \frac{\sum_{i=1}^{M} q_i Z_{ij}'(t)}{d_B^{+}(w)}\Big)P_j(w,t)dw\Big]\end{aligned} \tag{A·57}$$

$$\nabla\mathcal{L}_f(\Theta,T) = \frac{1}{M_Z T}\sum_{i=1}^{M} q_i Z_{ij}'(T) \tag{A·58}$$

References

Akkaya, K. and Younis, M. (2005). A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349.

Alamdari, S., Fata, E., and Smith, S. L. (2014). Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research*, 33(1):138–154.

Anta, A. and Tabuada, P. (2010). To sample or not to sample: Self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042.

Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2011). *The traveling salesman problem: a computational study*. Princeton University Press.

Arora, S. (1998). Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the Association for Computing Machinery (JACM)*, 45(5):753–782.

Astrom, K. and Bernhardsson, B. (2002). Comparison of riemann and lebesgue sampling for first order stochastic systems. *In Proceedings of the 41st IEEE Conference on Decision and Control*, 2:2011–2016.

Bansal, N., Blum, A., Chawla, S., and Meyerson, A. (2004). Approximation algorithms for deadline-tsp and vehicle routing with time-windows. *In Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 166–174.

Bayindir, L. (2016). A review of swarm robotics tasks. *Neurocomputing*, 172:292 – 321.

Bellingham, J. S., Tillerson, M., Alighanbari, M., and How, J. P. (2002). Cooperative path planning for multiple UAVs in dynamic and uncertain environments. *In Proceedings of IEEE Conference on Decision and Control*, pages 2816–2822.

Blackmore, L., Ono, M., and Williams, B. (2011). Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27(6):1080–1094.

Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., and Sudan, M. (1994). The minimum latency problem. *In Proceedings of the 26th annual ACM Symposium on Theory of Computing*, pages 163–171.

Bryson, A. E. and Ho, Y. C. (1975). *Applied optimal control: optimization, estimation and control*. CRC Press.

Bullo, F., Frazzoli, E., Pavone, M., Savla, K., and Smith, S. (2011). Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504.

Burgard, W., Moors, M., and Schneider, F. (2002). *Collaborative Exploration of Unknown Environments with Teams of Mobile Robots*, volume 2466 of *Lecture Notes in Computer Science*, book section 4, pages 52–70. Springer Berlin Heidelberg.

Cao, M., Morse, A., Yu, C., Anderson, B., and Dasgupta, S. (2011). Maintaining a directed, triangular formation of mobile autonomous agents. *Communications in Information and Systems*, 11(1):1–16.

Cardei, M., Thai, M. T., Li, Y., and Wu, W. (2005). Energy-efficient target coverage in wireless sensor networks. *In Proceedings of the 24th Annual IEEE International Conference on Computer and Communications*, pages 1976–1984.

Cassandras, C. G. (2014). The event-driven paradigm for control, communication and optimization. *Journal of Control and Decision*, 1(1):3–17.

Cassandras, C. G. and Lafortune, S. (2006). *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Cassandras, C. G. and Li, W. (2002). A receding horizon approach for solving some cooperative control problems. *In Proceedings of 49th IEEE Conference on Decision and Control*, 4:3760–3765.

Cassandras, C. G. and Li, W. (2005). Sensor networks and cooperative control. *European Journal of Control*, 11(4-5):436–463.

Cassandras, C. G., Lin, X., and Ding, X. (2013). An optimal control approach to the multi-agent persistent monitoring problem. *IEEE Transactions on Automatic Control*, 58(4):947–961.

Cassandras, C. G., Wardi, Y., Panayiotou, C. G., and Yao, C. (2010). Perturbation analysis and optimization of stochastic hybrid systems. *European Journal of Control*, 16(6):642 – 661.

Chakrabarty, K., Iyengar, S. S., Qi, H., and Cho, E. (2002). Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51(12):1448–1453.

Chandler, P. R., Pachter, M., and Rasmussen, S. (2001). UAV cooperative control. *In Proceedings of the 2001 American Control Conference*, 1:50–55.

Chang, C., Yu, G., Wang, T., and Lin, C. (2014). Path construction and visit scheduling for targets using data mules. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(10):1289–1300.

Chao, I. M., Golden, B. L., and Wasil, E. A. (1996a). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489.

Chao, I.-M., Golden, B. L., and Wasil, E. A. (1996b). Theory and methodology - the team orienteering problem. *European Journal of Operational Research*, 88:464–474.

Chen, J., Sun, D., Yang, J., and Chen, H. Y. (2010). Leader-follower formation control of multiple non-holonomic mobile robots incorporating a receding-horizon scheme. *International Journal of Robotics Research*, 29(6):727–747.

Chini, G., Poli, C., Oddi, G., and Pietrabissa, A. (2014). Receding horizon multi-vehicle routing for emergency scenarios. *In Proceedings of Mediterranean Conference on Control and Automation*, pages 374–379.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, DTIC Document.

Corke, P., Wark, T., Jurdak, R., Hu, W., Valencia, P., and Moore, D. (2010). Environmental wireless sensor networks. *In Proceedings of the IEEE*, 98:1903–1917.

Cortes, J., Martinez, S., Karatas, T., and Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255.

Cruz, D., McClintock, J., Perteet, B., Orqueda, O. A. A., Yuan, C., and Fierro, R. (2007). Decentralized cooperative control - a multivehicle platform for research in networked embedded systems. *IEEE Control Systems,*, 27(3):58–78.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Defoort, M., Kokosy, A., Floquet, T., Perruquetti, W., and Palos, J. (2009). Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A distributed receding horizon approach. *Robotics and Autonomous Systems*, 57(11):1094–1106.

Deittert, M., Richards, A., and Mathews, G. (2010). Receding horizon control in unknown environments: Experimental results. *In Proceedings of 2010 IEEE International Conference on Robotics and Automation*, pages 3008–3013.

Demir, O. and Lunze, J. (2012). Cooperative control of multi-agent systems with event-based communication. *In Proceedings of 2012 American Control Conference*, pages 4504–4509.

Desai, J., Kumar, V., and Ostrowski, J. (1999). Control of changes in formation for a team of mobile robots. *In Proceedings of the IEEE International Conference on Robotics and Automation*, 2:1556–1561.

Dimarogonas, D. V. and Johansson, K. H. (2009). Event-triggered cooperative control. *European Control Conference*, pages 3015–3020.

Dunbar, W. B. and Murray, R. M. (2006). Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42(4):549 – 558.

Earl, M. G. and D'Andrea, R. (2007). A decomposition approach to multi-vehicle cooperative control. *Robotics and Autonomous Systems*, 55:276–291.

Ekici, A. and Retharekar, A. (2013). Multiple agents maximum collection problem with time dependent rewards. *Computers and Industrial Engineering*, 64(4):1009 – 1018.

Fax, J. and Murray, R. (2004). Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476.

Finke, J., Passino, K. M., and Sparks, A. (2003). Cooperative control via task load balancing for networked uninhabited autonomous vehicles. *In Proceedings of 42nd IEEE Conference on Decision and Control*, pages 31–36.

Fowler, J. M. and D'Andrea, R. (2002). Distributed control of close formation flight. *In Proceedings of the 41st IEEE Conference on Decision and Control*, pages 2972–2977.

Frazzoli, E., Mao, Z. H., Oh, J. H., and Feron, E. (2001). Resolution of conflicts involving many aircraft via semidefinite programming. *Journal of Guidance, Control, and Dynamics*, 24(1):79–86.

Furukawa, T., Bourgault, F., Lavis, B., and Durrant-Whyte, H. F. (2006). Recursive bayesian search-and-tracking using coordinated uavs for lost targets. *In Proceedings of 2006 IEEE International Conference on Robotics and Automation*, pages 2521–2526.

Harmati, I. and Skrzypczyk, K. (2009). Robot team coordination for target tracking using fuzzy logic controller in game theoretic framework. *Robotics and Autonomous Systems*, 57(1):75–86.

Hart, J. K. and Martinez, K. (2006). Environmental sensor networks: A revolution in the earth system science. *Earth-Science Reviews*, 78(34):177–191.

Heemels, W., Sandee, J., and Van Den Bosch, P. (2008). Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590.

Hernandez-Perez, H. and Salazar-Gonzalez, J. J. (2014). The multi-commodity pickup-and-delivery traveling salesman problem. *Networks*, 63(1):46–59.

Izadi, H. A., Gordon, B. W., and Rabbath, C. A. (2012). Decentralized receding horizon control with communication bandwidth allocation for multiple vehicle systems. *Optimal Control Applications and Methods*, 33(1):1–22.

Jadbabaie, A., Lin, J., and Morse, A. S. (2003). Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001.

Ji, M. and Egerstedt, M. (2007). Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4):693–703.

Khazaeni, Y. and Cassandras, C. G. (2014). A new event-driven cooperative receding horizon controller for multi-agent systems in uncertain environments. *In Proceedings of IEEE 53rd Annual Conference on Decision and Control*, pages 2770–2775.

Khazaeni, Y. and Cassandras, C. G. (2015). An optimal control approach for the data harvesting problem. *In Proceedings of IEEE 54th Annual Conference on Decision and Control*, pages 5136–5141.

Klesh, A. T., Kabamba, P. T., and Girard, A. R. (2008). Path planning for cooperative time-optimal information collection. *In Proceedings of the American Control Conference*, pages 1991–1996.

Kushner, H. and Yin, G. (2003). *Stochastic Approximation and Recursive Algorithms and Applications*. Springer.

Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345 – 358.

Li, W. and Cassandras, C. G. (2003). Stability properties of a cooperative receding horizon controller. *In Proceedings of 42nd IEEE Conference Proceedings on Decision and Control*, 1:492–497.

Li, W. and Cassandras, C. G. (2006a). Centralized and distributed cooperative receding horizon control of autonomous vehicle missions. *Mathematical and computer modelling*, 43(9):1208–1228.

Li, W. and Cassandras, C. G. (2006b). A cooperative receding horizon controller for multivehicle uncertain environments. *IEEE Transactions on Automatic Control*, 51(2).

Lin, S. and Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516.

Lin, X. and Cassandras, C. G. (2015). An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces. *IEEE Transactions on Automatic Control*, 60(6):1659–1664.

Liu, M., Yang, Y., and Qin, Z. (2011). A survey of routing protocols and simulations in delay-tolerant networks. *Lecture Notes in Computer Science*, 6843:243–253.

Mayne, D. and Michalska, L. (1990). Receding horizon control of nonlinear systems. *IEEE Transactions Automatic Control*, 35(7):814–824.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

McLain, T., Chandler, P., Rasmussen, S., , and Pachter, M. (2001). Cooperative control of UAV rendezvous. *In Proceedings of American Control Conference*, pages 2309–2314.

Miskowicz, M. (2015). *Event-Based Control and Signal Processing*. CRC Press.

Moazzez-Estanjini, R. and Paschalidis, I. C. (2012). On delay-minimized data harvesting with mobile elements in wireless sensor networks. *Ad Hoc Networks*, 10:1191–1203.

Murphey, R. and Pardalos, P. M. (2002). *Cooperative control and optimization*, volume 66. Springer.

Murray, R. M. (2007). Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583.

Ny, J. L., Dahleh, M. A., Feron, E., and Frazzoli, E. (2008). Continuous path planning for a data harvesting mobile server. *In Proceedings of the IEEE Conference on Decision and Control*, pages 1489–1494.

Oh, K. K. and Ahn, H. S. (2014). Formation control and network localization via orientation alignment. *IEEE Transactions on Automatic Control*, 59(2):540–545.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. (1992). *Spatial Tessellations: Concepts and Applications of Voronoi*. Wiley, New York, NY, USA, 2nd edition.

Olfati-Saber, R. (2007). *Design of Behavior of Swarms: From Flocking to Data Fusion using Microfilter Networks*, pages 19–41. John Wiley and Sons, Ltd.

Panagou, D., Turpin, M., and Kumar, V. (2014). Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach. *In Proceedings of 2014 IEEE International Conference on Robotics and Automation*, pages 6757–6762.

Pandya, A., Kansal, A., and Pottie, G. (2008). Goodput and delay in networks with controlled mobility. *In Proceedings of IEEE Aerospace Conference*, pages 1–8.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications.

Parker, L. E. (1993). Designing control laws for cooperative agent teams. *In Proceedings of IEEE International Conference on Robotics and Automation*, pages C582–C587.

Pillac, V., Gendreau, M., Guret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

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

Pontryagin, L. S. (1987). *Mathematical theory of optimal processes*. CRC Press.

Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T. (2003). 3D conflict resolution of multiple aircraft via dynamic optimization. *In Proceedings of the AIAA Guidance, Navigation, and Control Conf. and Exhibit*.

Reinelt, G. (1991). TSPLIB: A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384.

Ren, W. (2006). Cooperative control design strategies with local interactions. *In Proceedings of 2006 IEEE International Conference on Networking, Sensing and Control*, pages 451–456.

Ren, W. and Beard, R. (2008). *Distributed consensus in multi-vehicle cooperative control: theory and applications*. Springer.

Ren, W., Beard, R. W., and Atkins, E. M. (2005a). A survey of consensus problems in multi-agent coordination. *In Proceedings of the 2005 American Control Conference*, pages 1859–1864.

Ren, W., Beard, R. W., and McLain, T. W. (2005b). Coordination variables and consensus building in multiple vehicle systems. *Cooperative Control*, 309:171–188.

Salz, N. P. (1965). A possible basis for a theory for traveling-salesman problem. *Operations Research*, S 13.

Salz, N. P. (1966). A theory for traveling salesman problem. *Operations Research*, S 14.

Schneider, J. J., Bukur, T., and Krause, A. (2010). Traveling salesman problem with clustering. *Journal of Statistical Physics*, 141(5):767–784.

Schwager, M., Rus, D., and Slotine, J.-J. (2009). Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research*, 28(3):357–375.

Shamma, J. S. (2007). *Cooperative control of distributed multi-agent systems*. Wiley Online Library.

Smith, R. N., Schwager, M., Smith, S. L., Rus, D., and Sukhatme, G. S. (2011). Persistent ocean monitoring with underwater gliders: Towards accurate reconstruction of dynamic ocean processes. *In Proceedings of 2011 IEEE International Conference on Robotics and Automation*, pages 1517–1524.

Sun, X., Cassandras, C. G., and Gokbayrak, K. (2014). Escaping local optima in a class of multi-agent distributed optimization problems: A boosting function approach. *In Proceeings of the IEEE 53rd Annual Conference on Decision and Control*, pages 3701–3706.

Tahbaz-Salehi, A. and Jadbabaie, A. (2008). A necessary and sufficient condition for consensus over random networks. *IEEE Transactions on Automatic Control*, 53(3):791–795.

Tang, H., Miller-Hooks, E., and Tomastik, R. (2007). Scheduling technicians for planned maintenance of geographically distributed equipment. *Transportation Research Part E: Logistics and Transportation Review*, 43(5):591 – 609.

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

Tanner, H. G., Jadbabaie, A., and Pappas, G. J. (2007). Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868.

Tekdas, O., Isler, V., Lim, J. H., and Terzis, A. (2009). Using mobile robots to harvest data from sensor fields. *IEEE Wireless Communications*, 16(1):22–28.

Tomasson, E. (2011). A receding horizon approach to resource allocation and data harvesting. Master's thesis, Boston University.

Trimpe, S. and D'Andrea, R. (2014). Event-based state estimation with variance-based triggering. *IEEE Transactions on Automatic Control*, 59(12):3266–3281.

Vansteenwegen, P., Souffriau, W., and Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10.

Wang, J. and Xin, M. (2013). Integrated optimal formation control of multiple unmanned aerial vehicles. *IEEE Transactions on Control Systems Technology*, 21(5):1731–1744.

Wei, W., Srinivasan, V., and Chua, K. C. (2008). Extending the lifetime of wireless sensor networks through mobile relays. *IEEE and Association for Computing Machinery Transactions on Networking*, 16(5):1108–1120.

Yamaguchi, H. and Arai, T. (1994). Distributed and autonomous control method for generating shape of multiple mobile robot group. *In Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2:800–807 vol.2.

Yao, C., Ding, X. C., and Cassandras, C. (2010). Cooperative receding horizon control for multi-agent rendezvous problems in uncertain environments. *In Proceedings of 49th IEEE Conference on Decision and Control*, pages 4511 –4516.

Yucelen, T. and Johnson, E. (2012). Cooperative control of uncertain multivehicle systems. *In Proceedings of the IEEE 51st Annual Conference on Decision and Control*, pages 837–842.

Zahn, C. T. and Roskies, R. Z. (1972). Fourier descriptors for plane closed curves. *IEEE Transactions on Computers*, C-21(3).

Zeng-Guang, H., Long, C., and Min, T. (2009). Decentralized robust adaptive control for the multiagent system consensus problem using neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,*, 39(3):636–647.

Zhao, W., Ammar, M., and Zegura, E. (2005). Controlling the mobility of multiple data transport ferries in a delay-tolerant network. *In Proceedings of the 24th IEEE International Conference on Computer and Communications*, 2:1407–1418.

Zhong, M. and Cassandras, C. G. (2010). Asynchronous distributed optimization with event-driven communication. *IEEE Transactions on Automatic Control*, 55(12):2735–2750.

Zhong, M. and Cassandras, C. G. (2011). Distributed coverage contorol and data collection with mobile sensor networks. *IEEE Transactions on Automatic Control*, 56(10):2445–2455.

CURRICULUM VITAE

YASAMAN KHAZAENI

Current Position: Research Staff Member, Cognitive User Experience Lab
IBM Research - Cambridge, MA

EDUCATION

PhD - Systems Engineering
Boston University - Boston, MA *September 2011-May 2016*
GPA: 4.0
M.Sc. - Petroleum Engineering:
West Virginia University - Morgantown WV *August 2007-December 2009*
Concentration: Reservoir Engineering, GPA: 4.0
B.Sc.- Petroleum Engineering/Reservoir Engineering
Sharif University of Technology, Tehran, IRAN *August 2003-June 2005*
B.Sc.- Electrical Engineering/Control Systems
Sharif University of Technology, Tehran, IRAN *August 2000-June 2005*

RESEARCH EXPERIENCE

CODES Lab - Boston University *September 2011-May 2016*
Control Of Discrete Event Systems Lab - *Boston, MA*

- Research in the area of cooperative multi-agent systems focusing on methods of optimal control, model predictive control, hybrid systems modeling and infinitesimal perturbation analysis..

IBM Research Collaboration with Boston University - Boston City Hall *June 2012*
Boston, MA

- Worked with traffic visualization group in the IBM Smarter City Challenge for city of Boston.

PEARL Lab - WVU *August 2007-April 2011*
Petroleum Engineering and Analytics Research Lab, *Morgantown, WV*

- Worked on several application of Artificial Intelligence in Reservoir modeling and prediction.

Brunel University *October 2005-August 2007*
London, UK

- Investigated the effect of Tollmien-Schlichting waves in plate boundary flow by developing a 2-D simulation model for Blasius equation.

WORK EXPERIENCE

Summer Intern: *May 2015 - August 2015*
Palo Alto Research Center (EAST), A Xerox Company

- Developed a novel inverse model for skin perfusion mapping and physiological parameter estimation using spectral and RGB imaging. (Pending US Patent)

Research Associate: *May 2010-August 2011*
West Virginia University, Morgantown, WV

- Supervised a team of graduate students working on a DOE funded project on "Monitoring/Verification/ Accounting(MVA), simulation, and risk assessment of CO_2 sequestration in geologic formations"

Teaching Instructor: *August 2010-August 2011*
West Virginia University, Morgantown, WV

- Advanced Reservoir simulation(Graduate level), Introduction to reservoir simulation (Graduate level)

Summer Intern: *May 2008 - August 2008*
Merrick Systems Co.

- Worked as a software designer and developer for a geostatistical analysis software.

AWARDS & HONORS

- Division of Systems Engineering Travel Award, Boston University. *2014, 2015*
- NSF Travel Award for the IEEE Conference on Decisions and Control. *2014*
- Dean's Fellowship for PhD Students in Systems Engineering, Boston University. *2011*
- Outstanding graduate student for academic achievement and excellence in research, West Virginia University. *2008, 2009*
- Invited discussion leader for the "Artificial Intelligence in the E&P industry Forum", Colorado Springs, June 2009. *2009*
- 2^{nd} prize in Nico van Wingen SPE annual scholarship. *2009*
- Soudavar scholarship (30,000 GBP) by Brunel University, London *2005*
- Three year undergraduate scholarship in Petroleum Engineering from Sharif University of Technology. *2003*
- 4^{th} place among over 450,000 participants in the national university entrance exam, IRAN. *2000*

SELECTED PUBLICATIONS

- Y. Khazaeni, C. G. Cassandras, *Event excitation for event-driven control and optimization of multi-agent systems*, 13th International Workshop on Discrete Event Systems, May 2016, Xi'an, China.
- Y. Khazaeni, C. G. Cassandras, *An Optimal Control Approach for the Data Harvesting Problem*, 2015 IEEE 54th Annual Conference on Decision and Control (CDC), Japan. pp. 5136-5141, Dec. 2015, Osaka, Japan.
- Y. Khazaeni, C. G. Cassandras, *Event-Driven Cooperative Receding Horizon Control for Multi-agent Systems in Uncertain Environments*, Under review in IEEE Transactions on Control and Network Systems.
- Y. Khazaeni, C. G. Cassandras, *A New Event-Driven Cooperative Receding Horizon Controller for Multi-agent Systems in Uncertain Environments*, 2014 IEEE 53rd Annual Conference on Decision and Control (CDC), pp.2770–2775, Dec. 2014, Los Angeles, CA.
- S.D. Mohaghegh, Y. Khazaeni, R. Gaskari, M. Maysami, *Data-Driven Reservoir Management of a Giant Mature Oilfield in the Middle East*, Society of Petroleum Engineers Annual Technical Conference and Exhibition (ATCE), 27 - 29 October 2014. RAI Centre Amsterdam, The Netherlands.
- S.D. Mohaghegh, Y. Al-Mehairi, R. Gaskari, M. Maysami, Y. Khazaeni, M. Gashut, A. E. Al-Hammadi and S. Kumar *Top-Down Modeling (TDM) of a Mature Giant Oilfield in the Middle East; Simultaneous History Matching of Production, Static Pressure and Time-Lapse Saturation* , ADIPEC 2013 Technical Conference, Abu Dhabi, UAE, 10-13 November 2013.
- Y. Khazaeni, S.D. Mohaghegh , *Intelligent Production Modeling Using Full Field Pattern Recognition*, SPE Reservoir Evaluation and Engineering-Reservoir Engineering, Volume 14, Number 6, December 2011, pp. 735-749
- Y. Khazaeni, S.D. Mohaghegh, *Intelligent Time-Successive Production Modelling*, SPE Western Regional Conference, Anaheim CA, May 2010. SPE 132643.