# Overcoming local optima in control and optimization of cooperative multi-agent systems

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

# OVERCOMING LOCAL OPTIMA IN CONTROL AND OPTIMIZATION OF COOPERATIVE MULTI-AGENT SYSTEMS

by

**SHIRANTHA WELIKALA**

B.Sc., University of Peradeniya, 2015
M.Sc., Boston University, 2020

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2021

Approved by

First Reader

Christos G. Cassandras, PhD
Distinguished Professor of Engineering
Professor of Electrical and Computer Engineering
Professor and Division Head of Systems Engineering

Second Reader

Sean B. Andersson, PhD
Professor of Mechanical Engineering
Professor of Systems Engineering

Third Reader

David A. Castañón, PhD
Professor of Electrical and Computer Engineering
Professor of Systems Engineering

Fourth Reader

Ioannis Ch. Paschalidis, PhD
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Computing and Data Sciences

*To my loving wife and caring parents.*

*If you subdue craving, your sorrows shall fall from you like drops of water from a lotus leaf. - Buddha, The Path of Dhamma, XXIV:3.*

## Acknowledgments

First, I would like to express my sincere gratitude to my advisor, Professor Christos G. Cassandras, for his continuous support, guidance, patience and trust throughout my doctoral studies. His insights and advice on research as well as on my personal growth have been invaluable. He has always been available, willing and prompt, whenever I needed help or advice. I feel incredibly privileged for having had the opportunity to learn from and do research with an excellent teacher and a well-renowned researcher like him over the past few years. I cannot even begin to imagine having a better advisor than him for my doctoral studies.

I would also like to thank my thesis committee members, Prof. Sean B. Andersson, Prof. David A. Castañón and Prof. Yannis Ch. Paschalidis, and, thesis committee chair, Prof. Hua O. Wang for their valuable time, thoughtful guidance, constructive comments and insightful questions regarding my thesis and research. It is an honor for me to have such a distinguished panel of researchers on my thesis committee.

I must thank my collaborators, Dr. Chuangchuang Sun and Samuel C. Pinto, for their intriguing ideas and hard work. Special thank goes to previous CODES lab members, Dr. Xinmiao Sun, Dr. Xiangyu Meng, Dr. Arian Houshmand, Dr. Nan Zhou and Dr. Rui Chen for their constant advice and help in my pursuit to extend their research contributions, and, Dr. Rebecca Swaszek and Dr. Yue Zhang for their friendly advice and support.

I extend my gratitude to the administrative staff in the Division of Systems Engineering and the Center for Information and Systems Engineering, Ms. Elizabeth Flagg, Ms. Christina Polyzos, Ms. Cheryl Stewart, Ms. Ruth Mason and Ms. Gabriella McNevin for their efficient handling of tedious administrative tasks, and, above all, for organizing events that created a friendly atmosphere in the division.

During my doctoral studies, I was fortunate to make friends with all the students who joined the Ph.D. program in the same year as me. Among them, I sincerely thank Wei Xiao and Salomon Wollenstein, who continued to join the CODES lab together with me. Seeing your passion for your research inspired me to do my best, and I immensely enjoyed our friendly conversations and occasional meals. I also like to thank Waleed Aslam, Kasra Ghasemi, Majid Heidarifar, Mahroo Bahranian, Ye Lin, Ricky, Yuping Wang and Jimmy Queeney for the fun conversations we had during various events and for the lengthy discussions we had to help each other out in numerous courses that we took together.

I express my gratitude to Gilbert Lepler, Leslee Rudnick, Margie Bleichman and Charlotte Craig for hosting me during the past few years in their cozy houses at 39 and 56 Thorndike Street, Brookline. Because of you, I got to celebrate Thanksgiving, Passover, Hanukkah, Christmas, birthdays and many other events (some even for no good reason!) each year. It immensely helped me to balance work and life.

I extend my gratitude to all teachers at Kingswood College, Sri Lanka and lecturers at the University of Peradeniya, Sri Lanka. I want to pay special regards to my previous research advisors, Dr. Roshan Godaliyadda, Dr. Parakrama Ekanayake, Prof. Janaka Ekanayake, Dr. Janaka Wijayakulasooriya and Dr. Lilantha Samaranayake who motivated as well as prepared me to pursue doctoral research. Special thanks go to my Advance Level teachers, Anil K. Senadheera, Keerthi Dharmasiri and Manoj Pathiraja for encouraging me to pursue perfection in my work and widening my horizon regarding future possibilities.

Last, certainly not least, I want to thank my dearest family, which starts with my loving wife, Iresha. During the past few years, even though we had to stay half a world away (literally!) from each other, I think, from heart and soul, we have been growing closer and closer to each other by the day. You constantly assisted me in any way that you could and helped me keeping track of my progress. I will always be indebted and thankful for your patience, support, faith and unconditional love. I am also immensely thankful to my mother, Seetha for giving me the courage to fight for what I desire, and, to my father, Sarath for teaching me the beauty of mathematics and engineering at a very young age. I will always be indebted for your unconditional love and for raising me to be the person I am today. I also want to thank my brothers, Sarantha and Sithara, my sister-in-law, Nadheesha, and, my dearest best friends, Vimukthi and Amith for their constant support and encouragement over the years. Finally, I like to thank my dogs, Rexy-I, Cindy, and Rexy-II, for their companionship. I could have never reached this achievement without my whole family behind me.

# OVERCOMING LOCAL OPTIMA IN CONTROL AND OPTIMIZATION OF COOPERATIVE MULTI-AGENT SYSTEMS

**SHIRANTHA WELIKALA**

Boston University, College of Engineering, 2021

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

## ABSTRACT

A cooperative multi-agent system is a collection of interacting *agents* deployed in a mission space where each agent is allowed to control its local state so that the fleet of agents collectively optimizes a common *global objective.* While *optimization* problems associated with multi-agent systems intend to determine the fixed set of globally optimal agent states, *control* problems aim to obtain the set of globally optimal agent controls. Associated non-convexities in these problems result in multiple local optima. This dissertation explores systematic techniques that can be deployed to either *escape* or *avoid* poor local optima while in search of provably better (still local) optima.

First, for multi-agent optimization problems with iterative gradient-based solutions, a *distributed* approach to escape local optima is proposed based on the concept of *boosting functions.* These functions temporarily transform gradient components at a local optimum into a set of *boosted* non-zero gradient components in a systematic manner so that it is more effective compared to the methods where gradient components are randomly perturbed. A novel variable step size adjustment scheme is also proposed to establish the convergence of this distributed boosting process. Developed boosting concepts are successfully applied to the class of coverage problems.

Second, as a means of avoiding convergence to poor local optima in multi-agent optimization, the use of *greedy algorithms* in generating effective *initial conditions* is explored. Such greedy methods are computationally cheap and can often exploit submodularity properties of the problem to provide performance bound guarantees to the obtained solutions. For the class of submodular maximization problems, two new performance bounds are proposed and their effectiveness is illustrated using the class of coverage problems.

Third, a class of multi-agent control problems termed *Persistent Monitoring on Networks* (PMN) is considered where a team of agents is traversing a set of nodes (targets) interconnected according to a network topology aiming to minimize a measure of overall node state. For this class of problems, a gradient-based parametric control solution developed in a prior work relies heavily on the initial selection of its 'parameters' which often leads to poor local optima. To overcome this initialization challenge, the PMN system's asymptotic behavior is analyzed, and an off-line greedy algorithm is proposed to systematically generate an effective set of initial parameters.

Finally, for the same class of PMN problems, a computationally efficient distributed on-line Event-Driven Receding Horizon Control (RHC) solution is proposed as an alternative. This RHC solution is *parameter-free* as it automatically optimizes its planning horizon length and *gradient-free* as it uses explicitly derived solutions for each RHC problem invoked at each agent upon each event of interest. Hence, unlike the gradient-based parametric control solutions, the proposed RHC solution does not force the agents to converge to one particular behavior that is likely to be a poor local optimum. Instead, it keeps the agents actively searching for the optimum behavior.

In each of these four parts of the thesis, an interactive simulation platform is developed (and made available online) to generate extensive numerical examples that highlight the respective contributions made compared to the state of the art.

# Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | | |
|---|---|---|
| ADMM | ............. | Alternating Direction Method of Multipliers |
| CBF | ............. | Control Barrier Functions |
| CBS | ............. | Centralized Boosting Scheme |
| DBS | ............. | Distributed Boosting Scheme |
| IPA | ............. | Infinitesimal Perturbation Analysis |
| KKT | ............. | Karush–Kuhn–Tucker |
| L.H.S. | ............. | Left Hand Side |
| MASE | ............. | Multi-Agent Simulation Example |
| MILP | ............. | Mixed Integer Linear Program |
| MPC | ............. | Model Predictive Control |
| OTCP | ............. | Optimal Threshold Control Policy |
| PMN | ............. | Persistent Monitoring on Networks |
| RHC | ............. | Event-Driven Receding Horizon Control |
| RHCP | ............. | Event-Driven Receding Horizon Control Problem |
| R.H.S. | ............. | Right Hand Side |
| RRHC | ............. | Reinforced Event-Driven Receding Horizon Control |
| SASE | ............. | Single-Agent Simulation Example |
| TBC | ............. | Threshold Based Control |
| TCEO | ............. | Target-Cycle Expansion Operation |
| TCP | ............. | Threshold Control Policy |
| TSP | ............. | Traveling Salesman Problems |
| w.r.t. | ............. | with respect to |

# Chapter 1

# Introduction

A cooperative multi-agent system is a collection of interacting subsystems (*agents*) where each agent controls its local state to collectively optimize a common *global objective* subjected to various *constraints*. Depending on the application, the agents of a multi-agent system may refer to sensors, vehicles, robots, service providers, or even processors. Also, the constraints faced by the agents can be thought of as a result of the given mission space and decision space limitations. Furthermore, the global objective can be thought of as a reward that depends on agent interactions with each other and with their surrounding mission space.

The optimal way to govern a multi-agent system is determined by a corresponding *optimization* (*static*) or *control* (*dynamic*) problem depending on the form of the seeking solution. In particular, an optimization problem aims to determine the fixed set of globally optimal agent states while a control problem seeks to determine the set of globally optimal agent controls (or equivalently, agent state-trajectories). Most of the optimization and control techniques found in the literature use *iterative solution update* schemes to obtain the optimal solution. Such an approach is called *distributed* if these updates can be executed at each agent separately and only using locally available information. Moreover, it is called *on-line* if these updates can be executed without having any initial or intermediate stages that use global information of the multi-agent system or the mission space.

## 1.1 Background and Motivation

Obtaining the globally optimal solution to an optimization or control problem associated with a multi-agent system is a challenging task depending on the nature of the involved: (i) agents, (ii) constraints, (iii) mission space, (iv) inter-agent interactions, and (v) global objective function. Therefore, a large number of optimization and control methods can be found in the literature specifically developed to address different classes of such problems.

For example, optimization and control of cooperative multi-agent systems arise in a wide variety of applications such as in coverage control (Zhong and Cassandras, 2011), resource allocation (Marden and Roughgarden, 2014), consensus (Sun et al., 2017a), learning (Xu et al., 2015), formation control (Lin et al., 2014), monitoring (Zhou et al., 2018), flocking (Ghapani et al., 2016). transportation (Dotoli et al., 2013), smart cities (Anagnostopoulos et al., 2018) and smart grid (Molzahn et al., 2017). In most of these applications, gradient-based iterative solution update schemes are typically used due to their *simplicity* (see the survey paper (Nedić and Liu, 2018)) to compute. However, more computationally complex schemes such as the Alternating Direction Method of Multipliers (ADMM) (Bastianello et al., 2018), Model Predictive Control (MPC) (Dai et al., 2017) and Control Barrier Functions (CBF) (Lindemann and Dimarogonas, 2019) are also gaining popularity in these application domains due to their greater *generality* within their respective scopes. For example, the ADMM method is applicable to optimization problems and can handle non-differentiable objective functions, noisy and asynchronous updates, and equality constraints (Bastianello et al., 2018; Boyd et al., 2010). Similarly, the CBF method is applicable to control problems and can handle non-linearities in the agent dynamics, noise in the system and measurements, and complex objective functions while also guaranteeing all constraints are not violated (Xiao et al., 2019).

However, when an iterative solution update scheme is used to solve a multi-agent optimization or control problem, achieving the global optimal solution highly depends on the *convexity* of the considered objective function and the nature of the feasible decision/control space of the problem. As an example, in multi-agent optimization, the aforementioned Relaxed-ADMM approach in (Bastianello et al., 2018) only converges to the global optimum when the objective function is convex. Further, in multi-agent control, the Threshold Control Policy (TCP) proposed in (Zhou et al., 2019) is unlikely to converge to the global optimum (within the class of such controllers) as the objective function is non-convex. Similarly, there are a large number of multi-agent systems where the objective function of interest is non-convex and thus has multiple local optima. This eventually hinders the process of attaining a globally optimal solution. The class of multi-agent coverage (optimization) problems (Zhong and Cassandras, 2011) and persistent monitoring (control) problems (Zhou et al., 2019) can serve as prime examples for this scenario.

In such non-convex situations, a generally applicable alternative is to use global optimization techniques such as simulated annealing (Kirkpatrick et al., 1983; Chiu and Lin, 2004), genetic algorithms (Holland, 1984; Davis, 1996), particle swarm algorithms (Kennedy and Eberhart, 1995; Yazdani et al., 2018) or ant colony optimization (Blum, 2005; Ilie and Bădică, 2013) (see also the survey papers (Floudas and Gounaris, 2008; Arora et al., 1995)). Note that these global optimization techniques are not limited to handling multi-agent optimization problems but also can be adopted to solve multi-agent control (i.e., dynamic) problems. For example, (Tfaili and Siarry, 2008) adopts probabilistic and meta-heuristic ant colony optimization concepts to solve a class of dynamic problems. Nevertheless, among these global optimization techniques, the salient feature that enables achieving the global optimality is the element of *randomness* introduced in the process of controlling agents.

Therefore, such techniques can be computationally intensive and usually infeasible for distributed and on-line optimization.

However, addressing the issue of non-convexity without compromising the computational simplicity by exploiting properties that the objective function or the multi-agent system may possess has recently attracted renewed attention for specific classes of multi-agent systems. Some examples are as follows. The concept of local optima smoothing introduced in (Addis et al., 2005) assumes that the given non-convex objective can be smoothed into a uni-modal function using a log-concave kernel. The approximate dual subgradient algorithm presented in (Zhu and Martínez, 2013) assumes Slater's condition and Strong Duality. The ladybug exploration method proposed in (Schwager et al., 2008) tries to hover over the probable local optima solutions aiming to find a better optimum.

Along the same lines, the balanced detection technique introduced in (Zhong and Cassandras, 2011) for coverage control problems focuses on changing the original objective function to encourage global exploration over local approximations to achieve a better optimum. A more structured approach of the same strategy is proposed in (Sun et al., 2014), which introduces the concept of *boosting functions* to escape local optima and seek better ones through an exploration of the search space exploiting the structural properties of the underlying multi-agent system. The greedy-gradient method proposed for a class of coverage control problems in (Sun et al., 2019) uses the submodularity property of the coverage objective to impose tight performance bound guarantees on the initial solution (generated via a *greedy algorithm*) - hence the final local optimal solution can be expected to deliver higher performance. The persistent monitoring problem in (Zhou et al., 2019) tries to abstract the multi-agent system representation originally used in (Lin and Cassandras, 2015) to overcome converging to locally optimal agent state trajectories. However, as will be elaborated in the sequel, these methods lack generality and also face significant limitations.

## 1.2 Multi-Agent Optimization Problems

In multi-agent optimization problems, when iterative solution update schemes are used, the two main avenues to overcome the issue of multiple local optima are by: (i) executing *local explorations* and (ii) finding proper *initializations*. Computationally intensive randomization based approaches for both of the above avenues have been extensively studied in the literature (see (Hoos and Stutzle, 2005) and (Lasdon and Plummer, 2008), respectively). In contrast, this thesis mainly focuses on approaches where the structural properties of the underlying problem are utilized - avoiding any form of randomization. Along these lines, the *boosting functions* method introduced in (Sun et al., 2014) and the *greedy initialization* method introduced in (Sun et al., 2019) respectively gives a local exploration technique and an initialization technique to address the issue of multiple local optima for a class of multi-agent coverage problems - exploiting its own structural properties. The objective of this class of coverage problems is to determine the best arrangement for a set of agents (sensors) in a given mission space to maximize the probability of detecting randomly occurring events over this mission space (see Fig. 1·1 for an application example). Note that this coverage problem framework is used in Chapters 2 and 3 of this thesis to highlight and validate the respective contributions made (to a much broader class of problems).

### 1.2.1 Distributed Boosting

The key idea behind the centralized boosting functions approach proposed in (Sun et al., 2014) is to temporarily alter the agent *local objective* functions whenever an *equilibrium* (i.e., a local optimum) is reached, by defining a set of *auxiliary local objective* functions. This process is carried out indirectly by systematically transforming each agent's *local gradient* into a new *boosted gradient*. Therefore, a "boosting

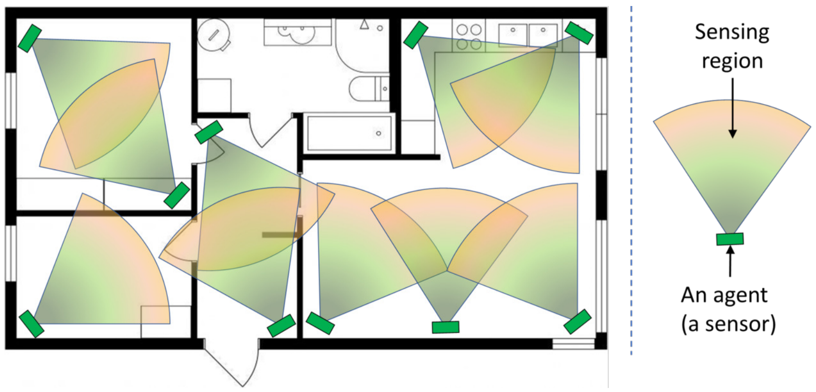


**Figure 1·1:** An application of the coverage problem: Determining the optimal arrangement for a set of agents (sensors) in a residential area.

function" is formally a transformation of local gradients into appropriate boosted gradients. Clearly, such transformations should always result in non-zero boosted gradients whenever the local gradients are zero, so as to facilitate escaping the local optima. After following the boosted gradients, when a new equilibrium point is reached, agents switch back to using local gradients (also called *normal gradients*). Subsequently, the gradient-based algorithm will converge to a new (potentially better) equilibrium point.

Compared to methods where gradient components are randomly perturbed to escape local optima (Kirkpatrick et al., 1983), the boosting function approach provides explicitly computed boosted gradients, which ensure both escaping from the local optima and subsequent systematic exploration of the search space. As will be shown, such desirable qualities can be achieved by designing boosted functions taking into account: (i) structural properties of the objective function, (ii) knowledge of feasible space, and (iii) agent state trajectories into account.

The boosting functions approach given in (Sun et al., 2014) is a centralized solution, and it has been established specifically to address the class of coverage problems introduced in (Zhong and Cassandras, 2011). Also, no convergence guarantees have been provided. Moreover, it uses only the structural properties of the objective function during its process of boosted gradient construction. These key limitations of (Sun et al., 2014) are addressed in Chapter 2 of this thesis - among making several other significant contributions (see also (Welikala and Cassandras, 2020a)).

### 1.2.2 Greedy Initialization

To overcome the issue of local optima faced in the class of coverage problems (Zhong and Cassandras, 2011), the work in (Sun et al., 2019) proposes a *greedy-gradient* algorithm. The underlying motivation is to exploit the *submodularity* property of the coverage objective function to determine a favorable initial condition for a subsequent gradient process using a computationally efficient greedy algorithm. The impact of having a submodular objective function is that it reveals a *performance bound* guarantee on the generated greedy solution with respect to the global optimal solution. As pointed out in (Sun et al., 2019), when the gradient ascent process is initialized with a greedy solution that preferably has a reasonable performance bound guarantee, it can be expected to converge to a better local optimum (compared to situations where random initialization is used).

Similar arguments have been the motivation behind incorporating a greedy initialization scheme to a variety of optimization problems across the spectrum. Some example applications are as follows: (i) for machine learning in the seminal paper on deep neural networks (Bengio et al., 2007), (ii) for K-means based consensus clustering in (Li and Liu, 2018), (iii) to solve Traveling Salesman Problems (TSP) in (Xie and Liu, 2009), (iv) to solve coverage problems as will be shown in Chapter 3 of this thesis, and (v) to solve persistent monitoring problems as will be shown in Chapter 4 of this thesis.

In particular, this thesis further improves the results established in (Sun et al., 2019) concerning the class of coverage problems, and, most importantly, makes a contribution to the general class of submodular maximization problems. In both these objectives, the primary focus is to establish improved performance bounds, as it allows one to put more confidence in the obtained solution regarding its closeness to the global optimal solution.

Formally, the performance bound of an obtained (greedy) solution is a lower bound to the ratio $f^G/f^*$ so that $\beta \leq f^G/f^*$, where $f^G$ and $f^*$ correspond to the objective function values under the obtained solution and the global optimal solution, respectively. It is shown to be $\beta = (1 - \frac{1}{e})$ when the objective function $f$ is *monotone submodular* and becomes $\beta = (1 - (1 - \frac{1}{N})^N)$ when the allowable maximum number of agents is constrained to $N$ (Fisher et al., 1978; Nemhauser et al., 1978). Note that having a performance bound closer to 1 is preferred as it yields that the obtained (greedy) solution is almost globally optimal.

The recent works related to *submodular maximization* problems have shown an increasing interest in improving upon the aforementioned *conventional* performance bounds by exploiting structural properties of the underlying problem. Specifically, these structural properties are defined by the nature of: (i) the objective function, (ii) the feasible space, and (iii) the generated greedy solution, of the considered submodular maximization problem. The typical approach is first to use one or a few of these factors to define a monotonicity metric (commonly known as a *curvature* measure) for the considered problem and then to develop an improved (closer to 1 compared to conventional counterparts) performance bound as a function of this curvature measure. Note that such an improved performance bound is established considering the same greedy solution and is preferred as it allows us to: (i) have a more accurate sense of proximity of the greedy solution to the optimality and (ii) make more informed decisions regarding spending extra resources to seek a better solution.

For example, the work in (Conforti and Cornuéjols, 1984) defines a curvature measure named *total curvature* based on the nature of the objective function and the feasible space for the general class of submodular maximization problems. Then, a provably improved performance bound is developed using the said total curvature measure. The authors of (Conforti and Cornuéjols, 1984) also propose another curvature metric named *greedy curvature* based on the generated greedy solution and use it to develop another performance bound. This same procedure is followed in (Wang et al., 2016) and (Liu et al., 2018) to propose new curvature metrics named *elemental curvature* and *partial curvature* respectively and then to develop improved performance bounds.

The work in (Sun et al., 2019) first proves that the coverage objective in (Zhong and Cassandras, 2011) is submodular. Then, it proposes a centralized greedy algorithm to determine an initial set of agent locations. Afterward, it exploits the established submodularity property to compute the aforementioned total curvature and elemental curvature metrics for the considered class of coverage problems and shows that the use of such metrics can improve the performance bound guarantees considerably. Finally, (Sun et al., 2019) proposes a greedy-gradient algorithm where the obtained greedy solution is used as the initial condition in a subsequent gradient process to reach a local optimum, and shows that this combined greedy-gradient approach is more effective compared to the centralized boosting approach proposed in (Sun et al., 2014). Chapter 3 of this thesis improves upon the aforementioned contributions made in (Sun et al., 2019) for the class of coverage problems and even establishes several new theoretical results for the general class of submodular maximization problems (see also (Sun et al., 2020)).

## 1.3 Multi-Agent Control Problems

The *optimization* problems introduced in the previous section are more commonly referred to as *parametric optimization* or *static optimization* problems as they intend to determine an optimal set of static parameters in a feasible space. In contrast, the *control* problems introduced in this section are also referred to as *non-parametric optimization* or *dynamic optimization* problems as they intend to determine an optimal set of functions (of time) in a function space.

In particular, multi-agent control problems aim to determine the optimal agent controls (or equivalently, the optimal agent state trajectories) in a paradigm where the objective function, the mission space, and the agents themselves behave in a time-dependent manner. A large number of solution techniques can be found in the literature that have been developed to address different sub-classes of such multi-agent control problems. Some commonly used solution techniques involve optimal control (Song et al., 2014), dynamic programming (Floudas et al., 1999), Lyapunov control (Wang et al., 2014), control barrier functions (Lindemann and Dimarogonas, 2019), reinforcement learning (Valenti, 2007), *receding horizon control* (Yao et al., 2010) and *parametric control* (Cassandras et al., 2010). Note that in parametric control, the control (non-parametric optimization) problem of interest is reduced to an optimization (parametric optimization) problem by parameterizing the agent control policy, i.e., by forcing the agents to select their continuous controls based on a static set of parameters.

Some related applications are as follows. Optimal dynamic formation control focusing on leader-follower networks is discussed in (Sun and Cassandras, 2016). The work in (Lan and Schwager, 2013) considers the trajectory planning problem (focusing periodic trajectories) for a sensing robot to estimate a time-changing Gaussian random field that exists in its surrounding environment. The problem of *distributed estimation* of a centralized linear system using a set of observers is considered in (Wang et al., 2019). The work in (Pinto et al., 2019) considers the problems of *distributed estimation* of a decentralized network system using a team of mobile agents (sensors). The class of *persistent monitoring* problems discussed in (Lin and Cassandras, 2015) attempts to control the agent motion so as to minimize an uncertainty metric associated with the given mission space. The work in (Zhou et al., 2019) tries to abstract some of the representations used in (Lin and Cassandras, 2015) aiming to find a better optimum.

As mentioned earlier, the second half of this thesis tackles explicitly the class of persistent monitoring problems formulated in (Zhou et al., 2019). In the following subsections, the related literature leading up to the work (Zhou et al., 2019), the proposed centralized off-line solution based on parametric control, and the proposed distributed on-line solution based on receding horizon control are introduced, respectively.

### 1.3.1 Persistent Monitoring on Networks (PMN) Problems

A persistent monitoring problem arises when a dynamically changing environment needs to be monitored by a set of agents who cannot adequately cover the environment if they remained stationary. This constraint of having to have non-stationary exploratory agents to cover the changing environment contrasts persistent monitoring problems from the (static) multi-agent coverage problems (Zhong and Cassandras, 2011). An example scenario where persistent monitoring is required is shown in Fig. 1·2 (notice the differences compared to the coverage application shown in Fig. 1·1).

Persistent monitoring problems have many applications across different domains such as in smart cities (Rezazadeh and Kia, 2019), transportation systems (Yamashita et al., 2003) and manufacturing plants (Liaqat et al., 2019) - where a team of agents can be used to monitor different regions of the environment for congestion, disruptions or any other dynamic events of interest. Further, in a smart grid (Caprari et al., 2010; Fan et al., 2018; Menendez et al., 2017), a team of agents can be used to inspect power plants and transmission lines. Additional applications include surveillance (Aksaray

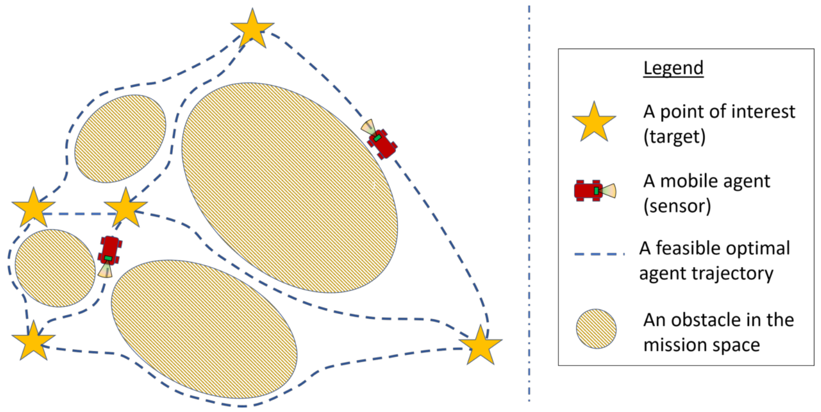

Legend
A point of interest (target)
A mobile agent (sensor)
A feasible optimal agent trajectory
An obstacle in the mission space

**Figure 1·2:** An example persistent monitoring problem setup. The number of agents are low and the agent sensing capabilities are limited (i.e., 'sensing range' values are small). Hence, mobile agents are required to regularly monitor the environment (points of interest).

et al., 2015; Maza et al., 2011), patrolling (Huynh et al., 2010), data collecting (Smith et al., 2011), sensing (Trevathan and Johnstone, 2018) and particle tracking (Shen and Andersson, 2011).

In general, persistent monitoring problems can be classified based on the nature of the *environment*, *objective* and *dynamics* involved. In particular, based on the nature of the environment, a monitoring problem may have a finite set of "points of interest" (Rezazadeh and Kia, 2019) or lack thereof (Maini et al., 2018) in the environment to be monitored. Based on the nature of the objective, different monitoring problems can be formulated to optimize event-counts (Yu et al., 2015), idle-times (Hari et al., 2019), error covariances (Pinto et al., 2020a) or visibility states (Maini et al., 2018) related to the environment. Finally, based on the nature of the environment dynamics, a monitoring problem can be either deterministic (Yu et al., 2016; Song et al., 2014) or stochastic (Rezazadeh and Kia, 2019; Lan and Schwager, 2013).

The persistent monitoring problem considered in this thesis (introduced in (Zhou et al., 2019)) focuses on an $n$-Dimensional ($n$-D) environment containing a finite number of points of interest (henceforth called "targets"). The agent team's objective is to collect information from (i.e., sense) each target to reduce an "uncertainty" metric associated with the target state. In particular, the dynamics of each target's uncertainty metric are such that it increases while no agent is present in the vicinity of the target and decreases when the target is being sensed by one or more agents in its vicinity. Therefore, the underlying global objective is to minimize an overall measure of target uncertainties by controlling the agent trajectories.

Persistent monitoring in 1-D environments has been addressed in (Zhou et al., 2018) by formulating an optimal control problem and showing that it can be reduced to a parametric optimization problem. This enables the use of Infinitesimal Perturbation Analysis (IPA) (Cassandras et al., 2010) to determine the gradients of the objective function with respect to the parameters and use gradient descent to determine their optimal values.

In contrast to the 1-D case, finding the solution to the problem of persistent monitoring in 2-D environments is much more complicated (Lin and Cassandras, 2015). However, as a remedy, the works in (Lin and Cassandras, 2015; Khazaeni and Cassandras, 2018a) propose to constrain agents to follow certain families of parametric trajectories (e.g., elliptical, Lissajous, Fourier) and use IPA to obtain an optimal solution within these families. However, as pointed out in (Zhou et al., 2019), limiting the agent trajectories to such forms can lead to poor local optima as such solutions cannot capture the dynamic changes in target uncertainties and highly depend on the initial target/agent conditions selected (Lin and Cassandras, 2015; Khazaeni and Cassandras, 2018a).

To overcome the challenges mentioned above, a graph topology is adopted in (Zhou et al., 2019) where the targets and the feasible inter-target agent trajectories are abstracted as graph nodes and edges, respectively. This abstraction has the added advantage of accounting for physical obstacles that might be present in the environment by constructing the graph accordingly (for example, see Fig. 1·3). In this persistent monitoring on networks (PMN) paradigm, an agent trajectory is fully characterized by a sequence of *visiting-targets* and the corresponding sequence of *dwell-times* to be spent at each visited target. Therefore, the controller that optimizes a given objective should yield such a (visiting-target, dwell-time) sequence for all agents. Clearly, this optimization problem is significantly more complicated than the NP-hard traveling salesman problem (Bektas, 2006) which only involves finding an optimal sequence of targets to visit. Thus, searching for the optimal (visiting-target, dwell-time) sequences for all the agents is a computationally-intensive process.

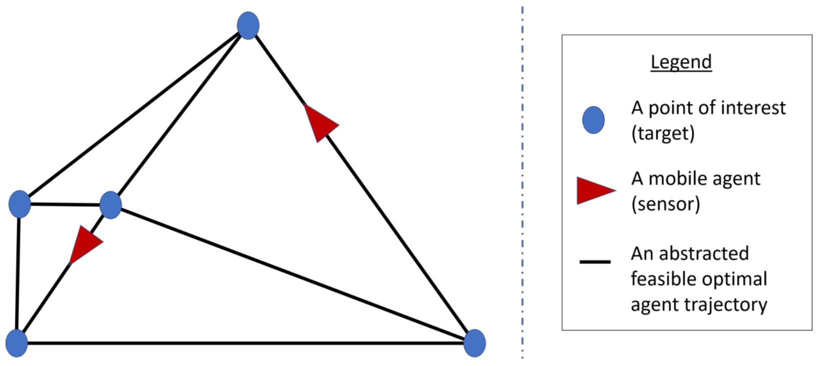

Legend
A point of interest (target)
A mobile agent (sensor)
An abstracted feasible optimal agent trajectory

**Figure 1·3:** The graph abstraction of the persistent monitoring problem setup shown in Fig. 1·2.

To overcome this issue, different PMN solutions in the literature have used different techniques. For example, (Rezazadeh and Kia, 2019) exploits the submodularity property of the objective function and proposes a sub-optimal greedy solution with a performance bound guarantee. The work in (Song et al., 2014) constrains agent trajectories to a closed path and optimizes agent speeds and initial locations. The work in (Yu et al., 2015) limits to a single-agent scenario and constrains the agent to a known cyclic visiting-target sequence to optimize the dwell-time sequence. A Mixed Integer Linear Program (MILP) is formulated in (Hari et al., 2019) to find the optimal cyclic visiting-target sequence limiting to a single-agent scenario. However, these approaches including many others (Maini et al., 2018; Hari et al., 2018) introduce additional constraints to the original PMN problem setup and are limited to centralized settings.

The work in (Zhou et al., 2019) overcomes these challenges by adopting a distributed threshold-based (parametric) control (TBC) approach where each agent enforces a set of thresholds on its neighboring target uncertainty values to decide immediate trajectory decisions (in a distributed manner): the dwell-time to be spent at the current target and the next target to visit. This parameterization enables the use of IPA to find optimal thresholds using a gradient descent process - in a distributed on-line manner.

### 1.3.2 Greedy Initialization for PMN Problems

Like all iterative solution update processes, the gradient-based threshold update process proposed in (Zhou et al., 2019) suffers from the issue of converging to a poor local optimum solution. This is due to the non-convexity of the persistent monitoring objective function of interest with respect to the thresholds. It is also important to note that such converged poor local optimum solutions are highly dependent on the initial thresholds used, which in (Zhou et al., 2019) are generated randomly.

Motivated by the previous use of *greedy initialization* in preventing iterative solution update processes from converging to poor local optima, Chapter 4 of this thesis appends a greedy initialization stage to the PMN solution proposed in (Zhou et al., 2019). In particular, this centralized off-line greedy initialization stage exploits global information available regarding the underlying network structure to determine a set of high-performing initial thresholds that ensures the subsequent IPA-based distributed on-line gradient descent process converges to an improved set of (still locally optimal) thresholds (see also (Welikala and Cassandras, 2019a)).

### 1.3.3 Event-Driven Receding Horizon Control for PMN Problems

While the developed TBC solution for PMN problems has many advantages like being simple and computationally efficient in the on-line phase, its need to have a centralized off-line stage can be seen as a disadvantage. Similarly, while having an on-line threshold-tuning component in the TBC solution can be seen as an advantage due to the offered adaptability, in certain applications, this might not be sufficient - in particular, when the agents have to directly react to various state (or system parameter) perturbations without having to go through a threshold-tuning process (with a considerable amount of recovery time) upon each such perturbation.

Motivated by these limitations, Chapter 5 of this thesis departs from the developed TBC solution and takes an entirely different approach to PMN problems. Specifically, the event-driven nature of PMN systems is exploited to derive an Event-Driven Receding Horizon Control (RHC) solution to optimally control each of the agents in a distributed on-line manner using only a minimal amount of computational power.

Compared to the TBC approach where agents may eventually converge to a certain stationary behavior characterized by the converged set of thresholds, in this RHC approach, agents continually keep on searching for the optimal behavior by globally optimizing a local version of the global objective function at each agent upon each event of interest. In that sense, this RHC approach shares some similarities with the earlier proposed *distributed boosting* approach for multi-agent optimization problems. For example, both are distributed on-line processes that intermittently motivate agents to search for the optimal agent states/behaviors. Moreover, as will be shown in Chapter 5, this RHC approach also has a *greedy initialization* component in it (see also (Welikala and Cassandras, 2021b)).

Before discussing the contributions of this thesis, a brief introduction to event-driven receding horizon control is provided in the following subsection.

### 1.3.4 A Brief Introduction to Event-Driven Receding Horizon Control

Often multi-agent control problems are complex, high-dimensional, and computationally intractable. When solving such problems, similar to using parametric control approaches, another feasible computationally efficient alternative is to use receding horizon control (also known as Model Predictive Control (MPC)). The motivation is to divide the main problem into a series of sub-problems that the agents can solve on-line in a distributed manner. Each such sub-problem aims to determine the optimal agent controls over a *planning horizon* that optimizes a local version of the interested global objective function. Upon solving such a sub-problem, the agent executes the determined optimal agent controls over a shorter *action horizon* defined either by the next time step or by the next event of interest that the agent observes. The same process is continued in this time-driven or event-driven manner, respectively.

In discrete-event or hybrid systems, event-driven receding horizon control (Li and Cassandras, 2006) can exploit the underlying event-driven nature of the system to reduce the computational complexity by orders of magnitude compared to time-driven receding horizon control (Dai et al., 2017) - due to its flexibility in the frequency of control updates. Typically, in RHC, an optimal control problem is solved upon each event of interest taking the current state of the system as the initial state to determine the subsequent optimal control actions. Hence RHC can provide an on-line solution compared to conventional control methods that use an off-line computed optimal control law defined over all states. In the literature, RHC has been used to address a wide range of cooperative multi-agent control problems such as multi-vehicle control (Li and Cassandras, 2006), multi-agent rendezvous (Yao et al., 2010), reward collection (Khazaeni and Cassandras, 2018a), and ride-sharing (Chen and Cassandras, 2020a).

## 1.4 Thesis Outline

In this thesis, Chapters 2, 3 are dedicated to multi-agent optimization problems while Chapters 4, 5 are dedicated to multi-agent control problems. In particular, Chapters 2 and 3 respectively extend the generality of the methods proposed in the aforementioned two works (Sun et al., 2014) and (Sun et al., 2019) while also overcoming their limitations by introducing several new concepts and establishing multiple new theoretical results. On the other hand, Chapters 4 and 5 focus primarily on the persistent monitoring on networks problem formulated in the aforementioned work (Zhou et al., 2019), and respectively develop (alternative) centralized off-line and distributed on-line solutions. It is important to point out that ongoing research has already adopted the novel techniques introduced in Chapters 4 and 5 of this thesis to address problems like distributed estimation (Pinto et al., 2019) as well as energy-aware multi-agent control (Bentz and Panagou, 2018) problems (see also (Welikala and Cassandras, 2020b) and (Welikala and Cassandras, 2021c)). Finally, Chapter 6 concludes this thesis by summarizing the main contributions and discussing ongoing and future research directions.

## 1.5 Contributions

The contributions of this dissertation are summarized as follows.

In Chapter 2, building upon the centralized boosting function approach introduced for the class of coverage problems in (Sun et al., 2014), a *distributed boosting function approach* is proposed to solve general non-convex optimization problems associated with cooperative multi-agent systems. In particular, first, a general-purpose Distributed Boosting Scheme (DBS) is proposed where each agent is allowed to asynchronously switch between a “boosting” and a “normal” mode independent of other agents and also without any global communication. Second, a generally applicable *optimal variable step size selection technique* is proposed to ensure the convergence of the DBS. Third, to provide more intuition about the boosting function design process (recall that a boosting function is formally a transformation of normal gradients into appropriate boosted gradients), for a class of multi-agent coverage problems (Zhong and Cassandras, 2011), two new boosting function families named *Arc-Boosting* and *V-Boosting* are developed. Finally, the effectiveness of the proposed distributed boosting function approach is illustrated by applying the developed DBS, the variable step size selection technique and the boosting function families to the said class of multi-agent coverage problems.

Chapter 3 explores the use of greedy initialization in multi-agent optimization problems. First, *two new performance bounds* are proposed for the general class of submodular maximization problems based on two new curvature metrics. Second, several important and useful properties related to the coverage objective function used in (Sun et al., 2019; Zhong and Cassandras, 2011) are established. Also, it is shown that the greedy algorithm proposed in (Sun et al., 2019) is inherently distributed. Finally, for this class of coverage problems, two performance bounds (based on *partial curvature* (Liu et al., 2018) and *greedy curvature* (Conforti and Cornuéjols, 1984)) taken from the literature along with the newly proposed two performance bounds are applied to obtain much-improved performance bounds compared to the those obtained in (Sun et al., 2019).

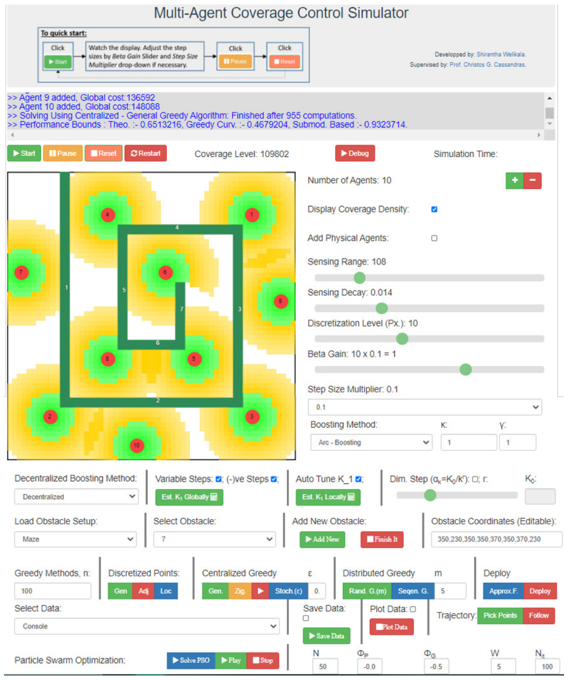
In Chapter 4, a greedy initialization technique is proposed for a class of parametric controllers deployed to solve persistent monitoring on networks problems. The contributions made in this chapter can be seen more broadly as a systematic approach to select effective initial conditions for gradient-based methods that solve non-convex optimization problems pertaining to a large class of dynamic multi-agent systems beyond persistent monitoring. In particular, this is accomplished by analyzing the asymptotic behavior of such systems and using the resulting optimal control policies to initialize a class of parametric controllers.

First, the *asymptotic analysis* of single-agent PMN systems is provided with the agent constrained to follow a periodic and non-overlapping sequence of targets (also called a "target-cycle"). Second, a *graph partitioning process* is proposed for multi-agent PMN systems to enable the extension of the said asymptotic analysis to deploy multiple agents. Third, a *computationally efficient, off-line greedy technique* is proposed to construct a high-performing set of thresholds for PMN problems. These thresholds are to be used as the initial thresholds in the subsequent on-line IPA-based gradient descent process. Finally, extensive simulation results are provided to show that these initial thresholds are often immediately optimal (still local) as well as much improved compared to the thresholds obtained in (Zhou et al., 2019). Thus, in such cases, the proposed greedy initialization scheme eliminates the need for any subsequent on-line gradient descent process.

Chapter 5 proposes an event-driven receding horizon control approach for PMN problems as an alternative to the parametric control approach developed in Chapter 4. The conventional use of RHC involves selecting a *planning horizon* over which each Event-Driven Receding Horizon Control Problem (RHCP) is solved (Li and Cassandras, 2006; Wang et al., 2017; Khazaeni and Cassandras, 2018a; Chen and Cassandras, 2020b). A novelty in the proposed RHC approach is *the ability to simultaneously determine the optimal value of the planning horizon* to be used locally at each agent at each RHCP. Moreover, an *explicit global optimal solution* is derived for each possible RHCP form, avoiding repetitive use of computationally intensive solvers or gradient-based methods. Therefore, the proposed RHC approach is parameter-free, computationally efficient as well as gradient-free.

In particular, first, it is shown that each agent's trajectory is fully characterized by the sequence of decisions it makes at specific discrete event times. Then, considering an agent at any one of these event times, an RHCP is formulated to determine the immediate optimal decisions over an optimally determined planning horizon. Second, several structural properties of this RHCP (which takes the form of a non-convex constrained optimization problem) are exploited to derive a unique global optimal solution for it in closed form. Third, some modifications are introduced to the proposed RHC architecture to obtain higher-performing solutions. Finally, the performance improvement achieved compared to the TBC solution proposed in (Zhou et al., 2019) and the controller's ability to take into account the presence of random effects affecting the system behavior are extensively studied using simulation experiments.

In each chapter of this thesis, an interactive simulation platform is developed (and made available online as shown in Fig. 1·4) to generate extensive numerical examples that highlight the respective contributions made compared to the state of the art.


Multi-Agent Coverage Control Simulator
To quick start:
Developped by: Shirantha Welikala.
Supervised by: Prof. Christos G. Cassandras.
Start
Pause
Reset
Restart
Debug
Simulation Time:

```
>> Agent 9 added, Global cost:136592
>> Agent 10 added, Global cost:148088
>> Solving Using Centralized - General Greedy Algorithm: Finished after 955 computations.
>> Performance Bounds : Theo. :- 0.6513216, Greedy Curv. :- 0.4679204, Submod. Based :- 0.9323714.
```

Coverage Level: 109802

Number of Agents: 10

Display Coverage Density: ✓

Sensing Range: 108

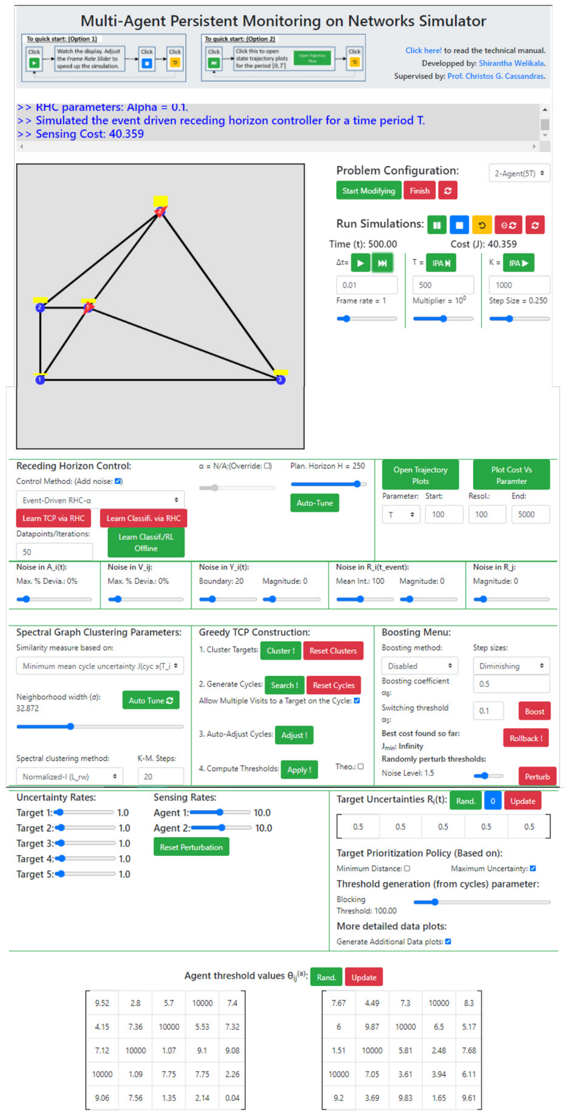
Sensing Decay: 0.014

Discretization Level (Px.): 10

Beta Gain: 10 x 0.1 = 1

Step Size Multiplier: 0.1

**(a)** Multi-agent coverage problem simulator (solver) used in Chapters 2 and 3 (available at: `http://www.bu.edu/codes/simulations/shiran27/CoverageFinal/`).


Multi-Agent Persistent Monitoring on Networks Simulator
To quick start: (Option 1)
To quick start: (Option 2)
Click here! to read the technical manual.
Developped by: Shirantha Welikala.
Supervised by: Prof. Christos G. Cassandras.
Problem Configuration:
Start Modifying
Finish
Run Simulations:
Receding Horizon Control:
Spectral Graph Clustering Parameters:
Greedy TCP Construction:
Boosting Menu:
Uncertainty Rates:
Sensing Rates:
Target Uncertainties R_i(t):
Rand.
Update

```
>> RHC parameters: Alpha = 0.1.
>> Simulated the event driven receding horizon controller for a time period T.
>> Sensing Cost: 40.359
```

Time (t): 500.00 Cost (J): 40.359

Agent threshold values $\theta_{ij}^{(a)}$:

| | | | | |
|---|---|---|---|---|
| 9.52 | 2.8 | 5.7 | 10000 | 7.4 |
| 4.15 | 7.36 | 10000 | 5.53 | 7.32 |
| 7.12 | 10000 | 1.07 | 9.1 | 9.08 |
| 10000 | 1.09 | 7.75 | 7.75 | 2.26 |
| 9.06 | 7.56 | 1.35 | 2.14 | 0.04 |

| | | | | |
|---|---|---|---|---|
| 7.67 | 4.49 | 7.3 | 10000 | 8.3 |
| 6 | 9.87 | 10000 | 6.5 | 5.17 |
| 1.51 | 10000 | 5.81 | 2.48 | 7.68 |
| 10000 | 7.05 | 3.61 | 3.94 | 6.11 |
| 9.2 | 3.69 | 9.83 | 1.65 | 9.61 |

**(b)** Multi-agent PMN problem simulator (solver) used in Chapters 4 and 5 (available at: `http://www.bu.edu/codes/simulations/shiran27/PersistentMonitoring/`).

**Figure 1·4:** Developed interactive JavaScript-based simulators used to validate the contributions of this thesis.

# Chapter 2

# Distributed Boosting for Multi-Agent Optimization

This chapter concentrates on the distributed boosting functions approach introduced in Section 1.2.1 which can be used to overcome the issue of multiple local optima arising in cooperative multi-agent optimization with non-convex objective functions. The key idea behind the boosting functions approach is to temporarily and systematically transform the local gradients used by the agents at a local optimum into a boosted gradient with a non-zero magnitude - to escape local optima. A distributed boosting scheme is developed along with a novel optimal variable step size selection mechanism to guarantee convergence of this DBS. Finally, simulation results are provided to demonstrate the effectiveness of the boosting function approach in attaining improved (still generally local) optima.

The sections of this chapter are arranged as follows. Section 2.1 introduces the general cooperative multi-agent optimization problem and the key concepts related to boosting functions approach. Then, an optimal variable step size selection mechanism along with related convergence proofs are provided in Section 2.2. In Section 2.3, the application of the boosting functions approach to the class of multi-agent coverage problems (Zhong and Cassandras, 2011) is discussed. Section 2.3.5 presents simulation results illustrating the effectiveness of the proposed distributed boosting framework and Section 2.4 concludes the chapter.

## 2.1 Problem Formulation

This chapter considers cooperative multi-agent optimization problems of the form

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} H(\mathbf{s}), \tag{2.1}$$

where, $H : \mathbb{R}^{mN} \rightarrow \mathbb{R}$ is the *global objective* function and $\mathbf{s} = [s_1, s_2, \ldots, s_N] \in \mathbb{R}^{mN}$ is the *global state*. Here, $s_i \in \mathbb{R}^m$ represents the controllable *local state* of an agent $i \in \{1, 2, \ldots, N\}$. The global objective function $H(\mathbf{s})$ is not required to satisfy any linearity or convexity-related conditions.

Inter-agent interactions are modeled by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of $N$ agents and $\mathcal{A}$ is the set of available communication links between those agents. The set of *neighbors* of an agent $i \in \mathcal{V}$ is denoted by $B_i = \{j : j \in \mathcal{V},\ (i, j) \in \mathcal{A}\}$ and the *closed neighborhood* of an agent $i$ is defined as $\bar{B}_i = B_i \cup \{i\}$. It is assumed that each agent $i$ shares its local state information $s_i$ with its neighbors in $B_i$. As a result, agent $i$ has knowledge of its *neighborhood state* $\bar{s}_i = \{s_j : j \in \bar{B}_i\}$.

Moreover, each agent (say $i$) is assumed to have a *local objective* function $H_i(\bar{s}_i)$ where $H_i : \mathbb{R}^{m|\bar{B}_i|} \rightarrow \mathbb{R}$ ($|\cdot|$ is the cardinality operator). Note that $H_i(\bar{s}_i)$ only depends on agent $i$'s neighborhood state $\bar{s}_i$. The relationship between local and global objective functions is not restricted to any specific form except for the condition:

$$\frac{\partial H_i(\bar{s}_i)}{\partial s_i} = 0,\ \forall i \in \mathcal{V} \implies \nabla H(\mathbf{s}) = 0. \tag{2.2}$$

This condition clearly holds for any problem with a *separable* form (Sun et al., 2014) $H(\mathbf{s}) = H_i(\bar{s}_i) + H_i^c(s_i^c)$ where $H_i^c : \mathbb{R}^{m(N-1)} \rightarrow \mathbb{R}$ and $s_i^c = [s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_N]$. Note that cooperative multi-agent systems which are inherently distributed (e.g., (Zhong and Cassandras, 2011)) naturally have separable objective functions. Moreover, many problems of interest with an *additive* form (Bastianello et al., 2018) $H(\mathbf{s}) = \sum_{i=1}^{N} H_i(\bar{s}_i)$ also satisfy this condition.

In order to solve (2.1), the *distributed* scheme where each agent $i$ updates its local state $s_i$ according to

$$s_{i,k+1} = s_{i,k} + \beta_{i,k} d_{i,k}, \tag{2.3}$$

is considered. Here, $\beta_{i,k} \in \mathbb{R}$ is a step size and $d_{i,k} \triangleq \frac{\partial H_i(\bar{s}_{i,k})}{\partial s_i} \in \mathbb{R}^m$ denotes the locally available gradient of agent $i$.

### 2.1.1 Escaping Local Optima Using Boosting Functions

The main idea behind the *boosting functions* approach is to temporarily replace the local objective function $H_i(\bar{s}_i)$ whenever an equilibrium is reached with an auxiliary objective function $\hat{H}_i(\bar{s}_i)$. Note that this is equivalent to replacing the *normal gradient* $d_i$ by a *boosted gradient* $\hat{d}_i = \frac{\partial \hat{H}_i(\bar{s}_i)}{\partial s_i}$ in (2.3).

#### Boosting Functions

A *boosting function* $f_i$ can be thought of as a transformation of an associated normal gradient $d_i$ which results in a boosted gradient $\hat{d}_i = f_i(d_i)$. In particular, this transformation takes place at an equilibrium point (where $d_i = 0$) and should result in a non-zero boosted gradient $\hat{d}_i = f_i(0) \neq 0$ which, therefore, forces agent $i$ to move in a direction determined by the boosting function and to explore the feasible space further. Subsequently, when a new equilibrium point is reached (i.e., when $\hat{d}_i = 0$), the agent reverts to the normal gradient $d_i$ and the gradient-based algorithm converges to a new (potentially better and never worse) equilibrium point.

The key to boosting functions is that they are selected to exploit the structure of: (i) the objective functions $H(\mathbf{s})$ and $H_i(\bar{s}_i)$, (ii) the gradient expression $d_i$, (iii) the feasible space and (iv) the agent state trajectory history. Therefore, unlike various forms of randomized state perturbations away from their current equilibrium

(Kirkpatrick et al., 1983; Chiu and Lin, 2004), boosting functions provide a formal rational systematic transformation process which depends heavily on the specific problem type.

**Boosting Function Example:** To provide insight into boosting functions in a generic setting, a general-purpose boosting function choice can be proposed as follows. In many multi-agent optimization problems, local optima arise when a cluster of agents provides a reasonably high performance by maintaining their local states in close proximity while completely ignoring globally dispersed state configurations. In such a case, a boosting function that enhances a separation among local states is a natural choice, especially suited for applications like coverage control, formation control, monitoring, consensus and transportation. Therefore, in a generic setting, a candidate boosted gradient $\hat{d}_i = f_i(d_i)$ for agent $i$ can be obtained by letting $\psi_{ij} = (s_i - s_j)$ and defining $\hat{d}_i = \nabla_{\psi_{ij}} H_i(\bar{s}_i)$ where its $l^{\text{th}}$ component is

$$\hat{d}_i^l = \frac{\partial H_i(\bar{s}_i)}{\partial \psi_{ij}^l} = \underbrace{\frac{\partial H_i(\bar{s}_i)}{\partial s_i^l}}_{=\, d_i^l} \frac{\partial s_i^l}{\partial \psi_{ij}^l} + \underbrace{\frac{\partial H_i(\bar{s}_i)}{\partial s_j^l}}_{\triangleq\, d_{ji}^l} \frac{\partial s_j^l}{\partial \psi_{ij}^l}. \tag{2.4}$$

Now, by replacing $\frac{\partial s_i^l}{\partial \psi_{ij}^l}$ and $\frac{\partial s_j^l}{\partial \psi_{ij}^l}$ with scalar parameters $\alpha_{ij}$ and $\eta_{ij}$, an entire *family of boosting functions* can be obtained as $\hat{d}_i = f_i(d_i) = \alpha_{ij} d_i + \eta_{ij} d_{ji}$ where $d_{ji} = \frac{\partial H_i(\bar{s}_i)}{\partial s_j}$ (see also (2.44) and (2.45)). Note that setting $\alpha_{ij} = 1$ and $\eta_{ij} = -1$ gives an interesting boosting function choice of the form $\hat{d}_i = f_i(d_i) = d_i - d_{ji}$. Since $d_{ji}$ represents the direction towards which agent $j$ should move to increase $H_i$, this is clearly an intuitive general choice for a boosting function at $i$. Details on selecting boosting functions along with some guidelines are provided in (Welikala and Cassandras, 2019b).

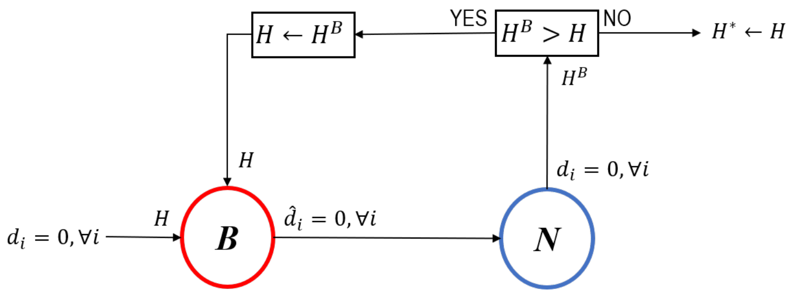


**Figure 2·1:** A centralized boosting scheme (CBS).

### Boosting Schemes

An agent $i$ is said to be in the *Boosting Mode* when it is following the boosted gradient direction $\hat{d}_i$ where its state updates take the form

$$s_{i,k+1} = s_{i,k} + \beta_{i,k}\hat{d}_{i,k}. \tag{2.5}$$

Similarly, when an agent $i$ is following the "normal" gradient direction $d_{i,k}$ as in (2.3), it is said to be in the *Normal Mode*. When developing an optimization scheme to solve (2.1), a proper mechanism, referred to as a *Boosting Scheme* is required to switch the agents between normal and boosting modes.

A centralized boosting scheme (CBS) is outlined in Fig. 2·1, where the boosting mode is denoted by $\boldsymbol{B}$ and the normal mode is denoted by $\boldsymbol{N}$. In a CBS, all agents are synchronized to operate in the same mode. In Fig. 2·1, $H$ denotes the global objective function value which is initially stored by all agents the first time mode $\boldsymbol{B}$ is entered when $d_i = 0$ for all $i \in \mathcal{V}$. After $\hat{d}_i = 0$ for all $i \in \mathcal{V}$, the agents re-enter mode $\boldsymbol{N}$ and, when a new equilibrium is reached, the new post-boosting value of the global objective function $H(\mathbf{s})$ is denoted by $H^B$. If $H^B > H$, an improved equilibrium point is attained and the process repeats by re-entering mode $\boldsymbol{B}$ with the new value $H^B$. The process is complete when this centralized controller fails to improve $H(\mathbf{s})$, i.e., when $H^B \leq H$.

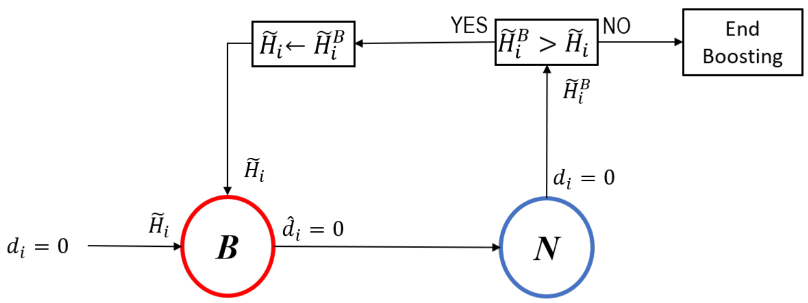


**Figure 2·2:** A distributed boosting scheme (DBS) asynchronously applied by each agent $i = 1, \ldots, N$.

This CBS was used in (Sun et al., 2014) with appropriately defined boosting functions in mode $\boldsymbol{B}$ to obtain improved performance for a class of multi-agent coverage problems. However, there has been no formal proof to date that this process converges. Moreover, the goal of this work is to develop a *Distributed Boosting Scheme* (DBS) where each agent can independently switch between modes $\boldsymbol{B}$ and $\boldsymbol{N}$ at any time. Such a scheme (i) improves the scalability of the system, (ii) eliminates the requirement of a centralized controller, (iii) reduces computational and communication costs, and, (iv) can potentially improve convergence times. Furthermore, this is a natural approach in problems such as coverage control (Zhong and Cassandras, 2011), where the original problem is inherently distributed.

A simple DBS version of Fig. 2·1 is shown in Fig. 2·2 where local use of the global objective $H$ is now replaced by a local estimate of $H$, denoted by $\bar{H}_i$, which will be formally introduced later. One can see that convergence of the DBS is far from obvious since agents may be at different modes at any time instant and, as their states change, the interaction among agents could lead to oscillatory behavior. Note that the notion of convergence involves not only the existence of equilibria such that $d_i = 0$ or $\hat{d}_i = 0$, but also a guarantee that the condition $H^B \leq H$ is eventually satisfied. It will be shown that the key to guaranteeing convergence is a process for *optimally selecting a variable step size* $\beta_{i,k}$ in (2.3) and (2.5).

**Convergence Criteria**

When a DBS is considered, the decentralized nature of agent behavior causes agents to switch between normal and boosting modes independently and asynchronously from each other (unlike a CBS). As a result, at a given time instant, a subset of the agents will be in normal mode (following (2.3)) while others are in boosting mode (following (2.5)). This creates a partition of the complete agent set $\mathcal{V}$ into two agent sets henceforth denoted by $\mathcal{N}$ and $\mathcal{B}$ respectively. Let us also define the *extended neighborhood* of an agent $i$ as $\tilde{B}_i \triangleq \cup_{j \in \bar{B}_i} \bar{B}_j$. For any agent $i \in \mathcal{V}$, the following conditions are defined as the *convergence criteria*:

$$\lim_{k\to\infty} d_{i,k} = 0 \text{ when } \tilde{B}_i \subseteq \mathcal{N}, \tag{2.6}$$

$$\lim_{k\to\infty} d_{i,k} = 0 \text{ when } i \in \mathcal{N}, \tilde{B}_i \cap \mathcal{B} \neq \emptyset, \tag{2.7}$$

$$\lim_{k\to\infty} \hat{d}_{i,k} = 0 \text{ when } i \in \mathcal{B}, \tilde{B}_i \cap \mathcal{B} \neq \emptyset. \tag{2.8}$$

These convergence criteria enforce the capability of an agent $i$ to escape its current mode (normal or boosting) irrespective of the surrounding neighbor mode partitions $\tilde{B}_i \cap \mathcal{N}$ and $\tilde{B}_i \cap \mathcal{B}$. Since boosting will only continue as long as there is a gain from the boosting stages (i.e., "$\bar{H}_i^B > \bar{H}_i$" in Fig. 2·2), it is clear how these criteria can lead all agents to terminate their boosting stages (i.e., to reach the "End Boosting" state). Finally, note that the criterion (2.6) applies to the convergence of any gradient-based method where boosting is not used.

## 2.2 Optimal Variable Step Sizes for Convergence

This section proposes a variable step size scheme which guarantees the convergence criteria (2.6)-(2.8) required when a general problem of the form (2.1) is solved using (2.3) and (2.5). The main results are Theorem 2.1 (which guarantees (2.6)) and

Theorem 2.2 (which guarantees (2.7) and (2.8)). These results depend on some assumptions which are presented first, starting with the nature of the local objective functions.

**Assumption 2.1.** *Any local objective function $H_i(\bar{s}_i)$, $i \in \mathcal{V}$, satisfies the following:*

1. *$H_i(\cdot)$ is continuously differentiable and its gradient $\nabla H_i(\cdot)$ is Lipschitz continuous (i.e., $\exists K_{1i}$ such that $\forall x, y \in \mathbb{R}^{m|\bar{B}_i|}$, $\|\nabla H_i(x) - \nabla H_i(y)\| \leq K_{1i}\|x - y\|$).*
2. *$H_i(\cdot)$ is a non-negative function with a finite upper bound $H_{UB}$, i.e., $H_i(x) < H_{UB} < \infty$, $x \in \mathbb{R}^{m|\bar{B}_i|}$.*

### 2.2.1 Convergence for Agents $i \in \mathcal{V}$ Such That $\tilde{B}_i \subseteq \mathcal{N}$

First, an optimal variable step size scheme is developed for agents $i \in \mathcal{V}$ such that $\tilde{B}_i \subseteq \mathcal{N}$, i.e., all agents in the extended neighborhood are in normal mode - following (2.3). The respective convergence criterion for this case is (2.6). For notational convenience, let $\mathbf{q}_i = \{1, 2, \ldots, q_i\}$ with $q_i = |\bar{B}_i|$ representing an ordered (re-indexed) version of the closed neighborhood set $\bar{B}_i$. For this situation, agent $i$'s neighborhood state update equation can be expressed as $\bar{s}_{i,k+1} = \bar{s}_{i,k} + \bar{\beta}_{i,k}\bar{\mathbf{d}}_{i,k}$ by combining (2.3) for all $j \in \bar{B}_i$. Here, $\bar{s}_{i,k+1}$, $\bar{s}_{i,k}$ and $\bar{\mathbf{d}}_{i,k}$ are $mq_i$-dimensional column vectors; equivalently, they may be thought of as $q_i \times 1$ block-column matrices with their $j^{\text{th}}$ block (of size $\mathbb{R}^{m\times 1}$, and $j \in \mathbf{q}_i$) being, $s_{j,k+1}$, $s_{j,k}$ and $d_{j,k}$ respectively. Accordingly, $\bar{\beta}_{i,k}$ is a $q_i \times q_i$ block-diagonal matrix, where its $j^{\text{th}}$ block on the diagonal (of size $m \times m$ and $j \in \mathbf{q}_i$) is $\beta_{j,k}I_m$; $I_m$ is the $m \times m$ identity matrix and $\beta_{j,k} \in \mathbb{R}$ is the step size of agent $j$.

The following lemma provides a modified version of the widely used descent lemma (Bertsekas, 2016) so that it can be applied to analyze maximization problems such as (2.1).

**Lemma 2.1.** *(Ascent lemma) For a function $f : \mathbb{R}^n \to \mathbb{R}$, if the Lipschitz continuity constant of $\nabla f$ is $L$, then, $\forall x, y \in \mathbb{R}^n$, $f(x + y) \geq f(x) + y^T\nabla f(x) - \frac{L}{2}\|y\|^2$.*

*Proof.* See Appendix A.2.1. □

Under Assumption 2.1, Lemma 2.1 can be applied to any local objective function $H_i(\bar{s}_{i,k})$ for the neighborhood state update $\bar{s}_{i,k+1} = \bar{s}_{i,k} + \bar{\beta}_{i,k}\bar{\mathbf{d}}_{i,k}$ as follows:

$$\begin{aligned} H_i(\bar{s}_{i,k+1}) &\geq H_i(\bar{s}_{i,k}) + (\bar{\beta}_{i,k}\bar{\mathbf{d}}_{i,k})^T \nabla H_i(\bar{s}_{i,k}) - \frac{K_{1i}}{2}\|\bar{\beta}_{i,k}\bar{\mathbf{d}}_{i,k}\|^2 \\ &= H_i(\bar{s}_{i,k}) + \sum_{j \in \bar{B}_i} \left[\beta_{j,k} d_{j,k}^T d_{ji,k} - \frac{K_{1i}}{2}\beta_{j,k}^2 \|d_{j,k}\|^2\right] \\ &= H_i(\bar{s}_{i,k}) + \sum_{j \in \bar{B}_i} \Delta_{ji,k}, \end{aligned} \tag{2.9}$$

with

$$\Delta_{ji,k} \triangleq \beta_{j,k} d_{j,k}^T d_{ji,k} - \frac{K_{1i}}{2}\beta_{j,k}^2 \|d_{j,k}\|^2 \in \mathbb{R}, \tag{2.10}$$

$$d_{ji,k} \triangleq \nabla_j H_i(\bar{s}_{i,k}) = \frac{\partial H_i(\bar{s}_{i,k})}{\partial s_j} \in \mathbb{R}^m. \tag{2.11}$$

Note that the term $d_{ji,k}$ in (2.11) is the sensitivity of agent $i$'s local objective $H_i$ to the local state $s_j$ of agent $j \in \bar{B}_i$. Also, $K_{1i}$ is the Lipschitz constant corresponding to $\nabla H_i$ and the term $\Delta_{ji,k}$ in (2.10) depends on the step size $\beta_{j,k}$ which is selected by agent $j \in \bar{B}_i$. In (2.9), each $\Delta_{ji,k}$ term can be thought of as a contribution from a neighboring agent $j$ to agent $i$, so as to improve (increase) $H_i$. The following assumptions ensures that any agent $i$ knows its contribution $\Delta_{ij,k}$ to an agent $j \in \bar{B}_i$.

**Assumption 2.2.** *Any agent $i \in \mathcal{V}$ has knowledge of the cross-gradient terms $d_{ij,k}$, the local Lipschitz constants $K_{1j}$, and the objective function values $H_j(\bar{s}_{j,k})$ at the $k^{th}$ update instant.*

This assumption is consistent with used concept of neighborhood, where neighbors share information through communication links.

Now a *neighborhood objective* function $\tilde{H}_i(\tilde{s}_{i,k})$ for any $i \in \mathcal{V}$, where $\tilde{H}_i : \mathbb{R}^{m|\tilde{B}_i|} \to \mathbb{R}$ and $\tilde{s}_{i,k} = \{s_j : j \in \tilde{B}_i\}$ is defined as follows:

$$\tilde{H}_i(\tilde{s}_{i,k}) = \sum_{j \in \bar{B}_i} H_j(\bar{s}_{j,k}). \tag{2.12}$$

This neighborhood objective function can be viewed as agent $i$'s estimate of the total contribution of agents in $\bar{B}_i$ towards the global objective function. These functions play an important role in the DBS because a distributed scheme comes at the cost of each agent losing the global information $H(\mathbf{s})$. Recall in a DBS, each agent $i$ uses a neighborhood objective function $\tilde{H}_i$ as a means of locally estimating the global objective function value (see "$\tilde{H}_i^B > \tilde{H}_i$" block in Fig. 2·2). However, as seen in the ensuing analysis, the form of $\tilde{H}_i$ is not limited to (2.12).

**Remark 2.1.** *In some problems, if the global and local objective functions are not directly related in an additive manner, then $\tilde{H}_i(\tilde{s}_{i,k}) = \sum_{j\in\bar{B}_i} w_{ij}H_j(\bar{s}_{j,k})$ can be used as a candidate for the neighborhood objective function. Here, $\{w_{ij} \in \mathbb{R}_{\geq 0} : j \in \bar{B}_i\}$ represents a set of weights (scaling factors). All results presented in this section can be generalized to such neighborhood objective functions as well.*

Enabled by the fact that $\tilde{B}_i \subseteq \mathcal{N}$, applying (2.9) to any agent $j \in \bar{B}_i$ gives $H_j(\bar{s}_{j,k+1}) \geq H_j(\bar{s}_{j,k}) + \sum_{l\in\bar{B}_j} \Delta_{lj,k}$. Summing both sides of this relationship over all $j \in \bar{B}_i$ and using the definition in (2.12) yields

$$\tilde{H}_i(\tilde{s}_{i,k+1}) \geq \tilde{H}_i(\tilde{s}_{i,k}) + (\tilde{\Delta}_{i,k} + Q_{i,k}), \tag{2.13}$$

where $\tilde{\Delta}_{i,k}$ and $Q_{i,k}$ are defined as

$$\tilde{\Delta}_{i,k} \triangleq \sum_{j\in\bar{B}_i} \Delta_{ij,k}, \tag{2.14}$$

$$Q_{i,k} \triangleq \sum_{j\in B_i} (\Delta_{jj,k} + \Delta_{ji,k} + \sum_{l\in B_j - \{i\}} \Delta_{lj,k}). \tag{2.15}$$

Note that $\tilde{\Delta}_{i,k}$ in (2.14) is a function of terms $\Delta_{ij,k}$ (and not $\Delta_{ji,k}$) which are locally available and controlled by agent $i$, i.e., via terms $\beta_{i,k}, d_{i,k}$ and $d_{ij,k}, \forall j \in \bar{B}_i$. However, agent $i$ does not have any control over $Q_{i,k}$ in (2.15), as this strictly depends (through (2.10)) on the step sizes of agent $i$'s neighbors in its *extended neighborhood* $\tilde{B}_i$ (i.e., $\beta_{j,k}, \forall j \in \tilde{B}_i - \{i\}$).

Nonetheless, (2.13) implies that the neighborhood objective function $\tilde{H}_i(\tilde{s}_{i,k})$ can be increased by at least $(\tilde{\Delta}_{i,k} + Q_{i,k})$ at any update instant $k$. Thus, to maximize the gain in $\tilde{H}_i(\tilde{s}_{i,k})$, agent $i$'s step size $\beta_{i,k}$ is selected according to the *auxiliary problem*:

$$\begin{aligned} \beta_{i,k}^* = \underset{\beta_{i,k}}{\arg\max} \quad & \tilde{\Delta}_{i,k} \\ \text{subject to} \quad & \tilde{\Delta}_{i,k} > 0. \end{aligned} \tag{2.16}$$

**Lemma 2.2.** *The solution to the auxiliary problem* (2.16) *is*

$$\beta_{i,k}^* = \frac{1}{\sum_{j\in\bar{B}_i} K_{1j}} \frac{d_{i,k}^T \sum_{j\in\bar{B}_i} d_{ij,k}}{\|d_{i,k}\|^2}. \tag{2.17}$$

*Proof.* Using (2.10) and (2.14), $\tilde{\Delta}_{i,k}$ can be written as

$$\tilde{\Delta}_{i,k} = \beta_{i,k} d_{i,k}^T \sum_{j\in\bar{B}_i} d_{ij,k} - \beta_{i,k}^2 \|d_{i,k}\|^2 \frac{\sum_{j\in\bar{B}_i} K_{1j}}{2}. \tag{2.18}$$

Note the quadratic and concave nature of $\tilde{\Delta}_{i,k}$ with respect to agent $i$'s step size $\beta_{i,k}$. Thus, using the KKT conditions (Bertsekas, 2016), the optimal $\beta_{i,k}$ can be directly obtained as (2.17). Let us denote the optimal objective function value as $\tilde{\Delta}_{i,k}^*$. It is easy to show that $\beta_{i,k}^*$ in (2.17) is feasible (i.e., $\tilde{\Delta}_{i,k}^* > 0$) as long as $\beta_{i,k}^* \neq 0$. $\square$

**Remark 2.2.** *The extreme situation where $\beta_{i,k}^* = 0$ occurs when $\sum_{j\in\bar{B}_i} d_{ij,k} = 0$. However, since this "pathological situation" can be detected by agent $i$, the agent can consider two options: (i) Use a reduced neighborhood $\bar{B}_i^1 \subset \bar{B}_i$ to calculate $\beta_{i,k}^*$ so that $\beta_{i,k}^* \neq 0$, hence $\tilde{\Delta}_{i,k}^* > 0$, or (ii) Use the weighted form of* (2.12) *(see Remark 2.1) and manipulate the weight factors $\{w_{ij} : j \in \bar{B}_i\}$ so as to get a step size $\beta_{i,k}^* \neq 0$ (e.g., enforcing $w_{ij} = 0, \forall j \ni d_{i,k}^T d_{ij} < 0$ will give $\beta_{i,k} > 0$, hence $\tilde{\Delta}_{i,k}^* > 0$). Therefore, in the following analysis, this pathological situation is omitted by assuming $\sum_{j\in\bar{B}_i} d_{ij,k} \neq 0$ (which implies $\beta_{i,k}^* \neq 0$).*

By substituting (2.17) in $\tilde{\Delta}_{i,k}$ given in (2.18), an explicit expression for $\tilde{\Delta}_{i,k}^*$ can be obtained as $\tilde{\Delta}_{i,k}^* = \frac{1}{2}\beta_{i,k}^* d_{i,k}^T \sum_{j\in\bar{B}_i} d_{ij,k}$. From this result and in view of Remark 2.2, it is clear that $\tilde{\Delta}_{i,k}^* \to 0$ if an only if $d_{i,k} \to 0$.

Next, regarding the term $Q_{i,k}$ in (2.15) over which agent $i$ does not have any control, the following lemma can be established.

**Lemma 2.3.** *The term $Q_{i,k}$ can be expressed as*

$$Q_{i,k} = \sum_{j \in B_i} (\tilde{\Delta}_{j,k} + \sum_{l \in B_j - \{i\}} [\Delta_{lj,k} - \Delta_{jl,k}]). \tag{2.19}$$

*Further, if $B_i = \bar{B}_j - \{i\}$, then under* (2.17)*, $Q_{i,k} > 0$.*

*Proof.* See Appendix A.2.2. □

**Assumption 2.3.** *Consider the sum,*

$$\tilde{Q}_{i,k} = \sum_{l=k-T_i}^{k} Q_{i,l}, \tag{2.20}$$

*such that $0 \leq T_i \leq k$. Then, $\exists T_i < \infty$ such that $\tilde{Q}_{i,k} \geq 0$.*

When the graph $\mathcal{G}(\mathcal{V}, \mathcal{A})$ is complete, the condition $B_i = \bar{B}_j - \{i\}$ in Lemma 2.3 is true for all $i \in \mathcal{V}$. In such cases, Assumption 2.3 is immediately satisfied with $T_i = 1, \forall i \in \mathcal{V}$. On the other hand, when the graph $\mathcal{G}(\mathcal{V}, \mathcal{A})$ is sparse enough, it can be considered as a collection of fully connected sub-graphs (exploiting the partitioned nature of local objective functions $H_i(\bar{s}_i)$). Then, Assumption 2.3 also holds with $T_i = 1, \forall i \in \mathcal{V}$. More generally, when each agent selects its step size according to (2.17), it ensures that $\tilde{\Delta}^*_{i,k} > 0$. In addition, $\Delta_{ii,k} > 0$ whenever the step size $\beta_{i,k}$ is positive. The assumption is further supported by the fact that each $Q_{i,k}$ in $\tilde{Q}_{i,k}$ is also a summation of $\Delta_{jj,k}$, $\Delta_{ji,k}$ and $\Delta_{lj,k}$ terms (noting in particular the positive first terms in (2.15) as well as in (2.19)). Moreover, it is locally verifiable if the agent communicates with its neighbors. In practice, this assumption has not been violated over extensive simulation examples (see Fig. 2·5 in Section 2.3).

**Assumption 2.4.** *For all $i \in \mathcal{V}$, there exists a function $\Psi_{i,k}$ and a finite positive* *number $\epsilon$ such that $\Psi_{i,k} \geq \epsilon > 0$ and*

$$\begin{cases} 0 \leq \Psi_{i,k}\|d_{i,k}\|^2 \leq \tilde{\Delta}^*_{i,k} + \tilde{Q}_{i,k}, \ \textit{when } \tilde{Q}_{i,k} > 0, \ \tilde{\Delta}^*_{i,k} > 0, & (2.21) \\ 0 \leq \Psi_{i,k}\|d_{i,k}\|^2 \leq \tilde{\Delta}^*_{i,k}, \ \textit{when } \tilde{\Delta}^*_{i,k} > 0. & (2.22) \end{cases}$$

This assumption is trivial because whenever the optimal step size in (2.17) is used, $0 < \tilde{\Delta}^*_{i,k}$, hence, for some $1 < K_2$, $\Psi_{i,k} = \tilde{\Delta}^*_{i,k}/(K_2\|d_{i,k}\|^2)$ is a candidate function for both cases (2.21) and (2.22) that satisfies the requirement $\Psi_{i,k} \geq \epsilon > 0$ for all $k$.

Now, the main convergence theorem can be established.

**Theorem 2.1.** *For all $i \in \mathcal{V}$ such that $\bar{B}_i \subseteq \mathcal{N}$, under Assumptions 2.1-2.4, the step size selection* (2.17) *guarantees the convergence criterion* (2.6)*, i.e.,* $\lim_{k\to\infty} d_{i,k} = 0$.

*Proof.* See Appendix A.2.3. □

### 2.2.2 Convergence for Agents $i \in \mathcal{V}$ Such That $\tilde{B}_i \cap \mathcal{B} \neq \emptyset$

In this case, at least some of the agents in $\tilde{B}_i$ are in boosting mode, following (2.5). Using the same approach as in Section 2.2.1, an optimal variable step size selection scheme is developed here so as to ensure the convergence criteria in (2.7) and (2.8).

Compared to (2.9), now the ascent lemma relationship for $H_i(\bar{s}_{i,k})$ takes the form:

$$H_i(\bar{s}_{i,k+1}) \geq H_i(\bar{s}_{i,k}) + \sum_{j\in\bar{B}_i\cap\mathcal{N}} \Delta_{ji,k} + \sum_{j\in\bar{B}_i\cap\mathcal{B}} \hat{\Delta}_{ji,k}, \tag{2.23}$$

where $\Delta_{ji,k}$ for $j \in \mathcal{N}$ is the same as (2.10) and

$$\hat{\Delta}_{ji,k} = \beta_{j,k}\hat{d}^T_{j,k}d_{ji,k} - \frac{K_{1i}}{2}\beta^2_{j,k}\|\hat{d}_{j,k}\|^2 \in \mathbb{R}. \tag{2.24}$$

Then, the ascent lemma for neighborhood objective function $\tilde{H}_i(\tilde{s}_{i,k})$ takes the form:

$$\tilde{H}_i(\tilde{s}_{i,k+1}) \geq \tilde{H}_i(\tilde{s}_{i,k}) + (\tilde{\Delta}_{i,k} + Q_{i,k}), \tag{2.25}$$

with (redefined) $\tilde{\Delta}_{i,k}$ and $Q_{i,k}$ as

$$\begin{aligned}
\tilde{\Delta}_{i,k} &\triangleq \mathbf{1}\{i \in \mathcal{N}\}[\sum_{j \in \bar{B}_i} \Delta_{ij(k)}] + \mathbf{1}\{i \in \mathcal{B}\}[\sum_{j \in \bar{B}_i} \hat{\Delta}_{ij(k)}], && (2.26)\\
Q_{i,k} &\triangleq \sum_{j \in B_i} (\mathbf{1}\{j \in \mathcal{N}\}[\Delta_{jj,k} + \Delta_{ji,k}] + \mathbf{1}\{j \in \mathcal{B}\}[\hat{\Delta}_{jj,k} + \hat{\Delta}_{ji,k}] && (2.27)\\
&\quad + \sum_{l \in \{B_j - \{i\}\}} [\mathbf{1}\{l \in \mathcal{N}\}\Delta_{lj,k} + \mathbf{1}\{l \in \mathcal{B}\}\hat{\Delta}_{lj(k)}]),
\end{aligned}$$

where $\mathbf{1}\{\cdot\}$ is the usual indicator function. Under this new $\tilde{\Delta}_{i,k}$ in (2.26), the same auxiliary problem as in (2.16) is used to determine the step size $\beta^*_{i,k}$ to optimally increase the neighborhood cost function $\tilde{H}_i(\tilde{s}_k)$.

**Lemma 2.4.** *The solution to the auxiliary problem* (2.16) *with* $\tilde{\Delta}_{i,k}$ *given in*(2.26) *is*

$$\beta^*_{i,k} = \begin{cases} \frac{1}{\sum_{j \in \bar{B}_i} K_{1j}} \frac{d^T_{i,k}(\sum_{j \in \bar{B}_i} d_{ij,k})}{\|d_{i,k}\|^2} & \text{when } i \in \mathcal{N}, \\ \frac{1}{\sum_{j \in \bar{B}_i} K_{1j}} \frac{\hat{d}^T_{i,k}(\sum_{j \in \bar{B}_i} d_{ij,k})}{\|\hat{d}_{i,k}\|^2} & \text{when } i \in \mathcal{B}. \end{cases} \tag{2.28}$$

*Proof.* The proof follows the same steps as that of Lemma 2.2 and is, therefore, omitted. □

Note that the step size selection criterion given in (2.28) (for an agent $i$) does not depend on its neighbors' modes. Therefore, it offers a generalization of (2.17). However, $\beta^*_{i,k}$ now depends on agent $i$'s own mode. This is due to the fact that the selection of $\beta^*_{i,k}$ allows agent $i$ to maximize the increment in the neighborhood objective function $\tilde{H}_i(\tilde{s}_i)$ which is defined in (2.12) independently from the boosting process. Therefore, the use of $\beta^*_{i,k}$ provides a regulation mechanism for the state update steps (especially during the boosting mode).

To establish the convergence criteria (2.7) and (2.8), Assumptions 2.1, 2.2 and 2.3 are still required. Note that Assumption 2.3 should now be considered under the new expression for $Q_{i,k}$ in (2.27). A generalized version of Lemma 2.3 is given as follows.

**Lemma 2.5.** *The term $Q_{i,k}$ in* (2.27) *can be expressed as,*

$$Q_{i,k} = \sum_{j \in B_i} (\tilde{\Delta}_{j,k} + \sum_{l \in B_j - \{i\}} [\mathbf{1}\{l \in \mathcal{N}\}\Delta_{lj,k} - \mathbf{1}\{j \in \mathcal{N}\}\Delta_{jl,k} + \mathbf{1}\{l \in \mathcal{B}\}\hat{\Delta}_{lj,k} - \mathbf{1}\{j \in \mathcal{B}\}\hat{\Delta}_{jl,k}]). \tag{2.29}$$

*Further, if $B_i = \bar{B}_j - \{i\}$, then under* (2.28)*, $Q_{i,k} > 0$.*

*Proof.* The proof follows the same steps as that of Lemma 2.3 and is, therefore, omitted. □

Finally, Assumption 2.4 needs to be modified as follows (to incorporate $i \in \mathcal{B}$).

**Assumption 2.5.** *For all $i \in \mathcal{V}$, there exists a function $\Psi_{i,k}$ and a finite positive number $\epsilon$ such that $\Psi_{i,k} \geq \epsilon > 0$ and,*
*if $i \in \mathcal{N}$:*

$$\begin{cases} 0 \leq \Psi_{i,k}\|d_{i,k}\|^2 \leq \tilde{\Delta}^*_{i,k} + \tilde{Q}_{i,k} & \text{when } \tilde{\Delta}^*_{i,k} > 0,\ \tilde{Q}_{i,k} > 0, \\ 0 \leq \Psi_{i,k}\|d_{i,k}\|^2 \leq \tilde{\Delta}^*_{i,k} & \text{when } \tilde{\Delta}^*_{i,k} > 0, \end{cases}$$

*otherwise, if $i \in \mathcal{B}$:*

$$\begin{cases} 0 \leq \Psi_{i,k}\|\hat{d}_{i,k}\|^2 \leq \tilde{\Delta}^*_{i,k} + \tilde{Q}_{i,k} & \text{when } \tilde{\Delta}^*_{i,k} > 0,\ \tilde{Q}_{i,k} > 0, \\ 0 \leq \Psi_{i,k}\|\hat{d}_{i,k}\|^2 \leq \tilde{\Delta}^*_{i,k} & \text{when } \tilde{\Delta}^*_{i,k} > 0. \end{cases}$$

*Here, $\tilde{Q}_{i,k}$ is evaluated from* (2.20) *using* (2.27) *and, $\tilde{\Delta}^*_{i,k}$ from* (2.26) *using* (2.28)*.*

The following convergence theorem can now be established.

**Theorem 2.2.** *Under Assumptions 2.1-2.3, and 2.5, the step size selection in* (2.28) *guarantees the convergence conditions stated in* (2.6)-(2.8)*: if $i \in \mathcal{N}$, then $\lim_{k\to\infty} d_{i,k} = 0$, and, if $i \in \mathcal{B}$, then $\lim_{k\to\infty} \hat{d}_{i,k} = 0$.*

*Proof.* See Appendix A.2.4. □

### 2.2.3 Discussion

**Feasible Space Constraint:** When the main problem in (2.1) includes a feasible space constraint $\mathbf{s} \in \mathbf{F} \subset \mathbb{R}^{mN}$, the standard gradient projections (Bertsekas, 2016) can be used for (2.3) and (2.5). For such a situation, the following lemma presents an additional condition which needs to be satisfied in order to guarantee the convergence of the proposed variable step size method (2.28).

**Lemma 2.6.** *If feasible space* $\mathbf{F}$ *is convex, and if an agent* $i$*'s local and cross gradients satisfy the conditions,*

$$\begin{aligned} |d_{i,k}^T \sum_{j\in B_i} d_{ij,k}| &< \|d_{i,k}\|^2 \text{ when } i \in \mathcal{N}, \\ |\hat{d}_{i,k}^T(\sum_{j\in B_i} d_{ij,k} + (d_{i,k} - \hat{d}_{i,k}))| &< \|\hat{d}_{i,k}\|^2 \text{ when } i \in \mathcal{B}, \end{aligned} \tag{2.30}$$

*the step sizes* $\beta_{i,k} = \beta_{i,k}^*$ *given by* (2.28) *when used in* (2.3) *or* (2.5) *with standard gradient projections (onto* $\mathbf{F}$*), will lead the state* $s_{i,k}$ *to a stationary point.*

*Proof.* See Appendix A.2.5. □

From a practical standpoint, if the conditions in Lemma 2.6 are being violated during the gradient ascent process, the neighborhood reduction and/or weight factor manipulation techniques mentioned in Remark 2.2 can be used to change $B_i$ and/or $\tilde{H}_i$ respectively so that these conditions are satisfied. Note also that the knowledge of the feasible space constraint $\mathbf{s} \in \mathbf{F}$ in (2.1) can play an important role in designing boosting functions $\hat{d}_i = f_i(d_i, \mathbf{F})$, as further discussed in Section 2.3.

**Variable Step Sizes Vs Fixed Step Sizes:** In a centralized setting, using a fixed step size for the gradient ascent is typically computationally inexpensive, and, if properly executed, can deliver a higher convergence rate compared to variable step size methods. However, in a distributed setting where agents independently and asynchronously alter the gradient direction ((2.3) and (2.5)), using a fixed step size (typically $\beta_{i,k} = \frac{1}{K_i}$) might not lead to good overall convergence properties. Further, establishing convergence in this case generally requires additional restrictive assumptions. In contrast, the proposed variable step size method has the following advantages: (i) It is designed so as to account for the distributed and cooperative nature of the underlying problem, (ii) Its convergence has been established by making only a few locally verifiable assumptions, (iii) It is not computationally heavy compared to line search methods, and, (iv) During different modes (boosting/normal) the step sizes are automatically adjusted. As a result, the variable step size method in applications has shown better convergence results compared to fixed step size methods (see Sections 2.2.4 and 2.3).

**Escaping and converging to saddle points:** Due to the non-convexity of the objective function, saddle points may exist in the feasible space. However, as shown in (Lee et al., 2017; Panageas et al., 2019), first-order methods (2.3) almost always avoid a large class of saddle points (called strict saddle points) inherently. Nevertheless, if boosting functions are deployed through (2.5), clearly, saddle points are easier to escape from compared to local minima. Moreover, even if the convergence criteria (2.6) - (2.8) lead to a saddle point, it will have a higher cost compared to initially attained local minima (or saddle points) as a result of the comparison stage used in boosting schemes (e.g., see "$H^B > H$" block in Fig. 2·1).

### 2.2.4 An Example for the Variable Step Size Method

In this section, a simple example is provided to illustrate the operation and convergence (i.e., validity) of the proposed variable step size method. In this example, the local objective functions are restricted to take a quadratic form:

$$H_i(\bar{s}_i) = -\Big\| \sum_{j\in\bar{B}_i} A_{ij}s_j - b_i \Big\|_{C_i}^2 = -\|g_i(\bar{s}_i)\|_{C_i}^2, \tag{2.31}$$

where $A_{ij} \in \mathbb{R}^{r\times m}, b_i \in \mathbb{R}^r$ and $C_i \in \mathbb{R}^{r\times r}$ for any $i \in \mathcal{V}, j \in \bar{B}_i$. The weight matrix $C_i$ is symmetric and positive definite. The weighted norm is defined as $\|v\|_C^2 = v^T C v$ with $v \in \mathbb{R}^r$ and $C \in \mathbb{R}^{r\times r}$. The parameter $r$ represents the dimension of the local cost function. Assuming the parameters $A_{ij}, b_i, C_i,\ \forall i \in \mathcal{V},\ \forall j \in \bar{B}_i$ and the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ are predefined (for a given $N, m$ and $r$ value combination), the considered optimization problem is

$$\mathbf{s}^* = [s_1^*, s_2^*, \ldots, s_N^*] = \arg\max_{\mathbf{s}} H(\mathbf{s}) = \sum_{i=1}^{N} H_i(\bar{s}_i). \tag{2.32}$$

Due to the quadratic nature of the associated objective functions, a closed form expression can be obtained for the global optimum $\mathbf{s}^*$. Moreover, as a result of convexity, there is no need for any boosting function to escape an equilibrium point. Therefore, this example is used to compare the performance of the proposed variable step size method (when used in a distributed gradient ascent) to that of a fixed step size method (when used in a centralized gradient ascent).

For the (distributed) variable step size computation (at agent $i$ via (2.17)), the local gradient $d_i$, cross gradients $d_{ij}, j \in \bar{B}_i$, and Lipschitz constants $K_{1j}, j \in \bar{B}_i$ are

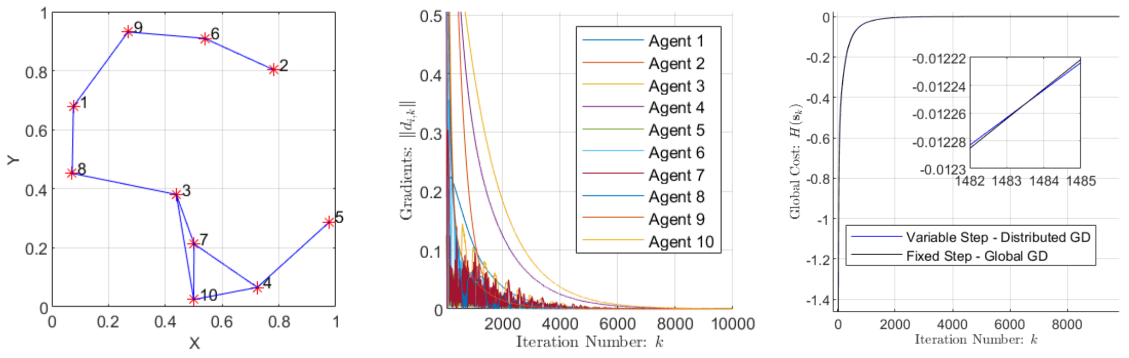
$$\begin{aligned} d_i &= \frac{\partial H_i(\bar{s}_i)}{\partial s_i} = -2A_{ii}^T C_i g_i(\bar{s}_i), && (2.33)\\ d_{ij} &= \left[\frac{\partial H_j(\bar{s}_j)}{\partial s_i}\right]_{i \leftrightarrow j} = -2A_{ji}^T C_j (\sum_{l \in \bar{B}_j} A_{jl} s_l - b_j), && (2.34)\\ K_{1j} &= 2\|A_j^T C_j A_j\|_\infty,\ A_j = [\{A_{jl}\}_{l \in \bar{B}_j}] \in \mathbb{R}^{r \times m|\bar{B}_j|}, && (2.35) \end{aligned}$$

respectively. However, in centralized gradient ascent, the global gradient of agent $i$:

$$d_i^G = \frac{\partial H(\mathbf{s})}{\partial s_i} = -2\sum_{j \in \bar{B}_i} A_{ji}^T C_j g_j(\bar{s}_j), \tag{2.36}$$

is used as a replacement for $d_{i,k}$ in (2.3). In this case, the step size is kept fixed at $\frac{1}{K_i}$ where $K_i = \sum_{j \in \bar{B}_i} K_{1j}$ (see (Bertsekas, 2016)).

In simulations, fixed dimensional parameters $N = 10$ and $m = r = 2$ are used. Note that $m = r$ is required here to guarantee the existence of a solution where $d_i = 0, \forall i \in \mathcal{V}$. It it is easy to show that the optimal global objective function value



**(a)** Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. **(b)** Local Derivatives. **(c)** Global Objective.

**Figure 2·3:** Numerical Example.

is $H(\mathbf{s}^*) = 0$. To generate the inter-agent connections (i.e., the graph $\mathcal{G}$) a random geometric graph generation is used taking 0.4 as the communication range parameter (Bastianello et al., 2018). The remaining problem parameters $A_{ij}, b_i, C_i, s_{i,0}$ $\forall i \in \mathcal{V}$, $\forall j \in \bar{B}_i$ are generated randomly (keeping the graph $\mathcal{G}$ fixed).

The experimental results shown in Fig. 2·3 confirm the established theoretical results. The $H(\mathbf{s}_k)$ profiles in Fig. 2·3(c) show that the proposed distributed variable step size method provides a slightly faster convergence than the centralized fixed step size method for $k \leq 1483$ where at $k = 1483$, the $H(\mathbf{s}_k)$ value is 99.95% closer to the optimal than the initial value $H(\mathbf{s}_0) = 26.1432$; for $k \geq 1484$, the centralized fixed step method is slightly faster. This cross-over behavior can be understood as a result of local gradients $d_{i,k}$ becoming smaller as $k$ increases and adapting step sizes $\beta_{i,k}$ in (2.3) when $d_{i,k}$ is very small is less effective. The general observation over extensive similar examples is that the result of such a comparison (between distributed variable step and centralized fixed step methods) depends on the network topology.

## 2.3 Application to Multi-Agent Coverage Problems

This section first introduces the class of multi-agent coverage problems considered in (Zhong and Cassandras, 2011; Sun et al., 2014). Then, two new boosting function families are specifically designed for this class of problems and applied in convergence guaranteed DBS following the theory developed in previous sections.

### 2.3.1 Multi-Agent Coverage Problem Formulation

The coverage problem aims to find an optimal arrangement for a given set of sensor nodes (agents) inside a given *mission space* so as to maximize the probability of detecting randomly occurring events (in the mission space).

The mission space $\Omega \subset \mathbb{R}^2$ is modeled as a non-self-intersecting polygon (Zhong and Cassandras, 2011) and it may contain a finite set of non-self-intersecting polygonal obstacles denoted by $\{M_1, M_2, \ldots, M_h\}$, where $M_i \subset \mathbb{R}^2$ represents the interior space of the $i^{\text{th}}$ obstacle. Therefore, agent motion and deployment are constrained to a non-convex feasible space $F = \Omega \backslash (\cup_{i=1}^{h} M_i)$. Note that "$\backslash$ "denotes the set subtraction operator. The spacial likelihood of random event occurrence over the mission space is quantified by the *event density* function $R : \Omega \to \mathbb{R}$, where, $R(x) \geq 0, \forall x \in \Omega$; $R(x) = 0$, $\forall x \notin F$, and $\int_{\Omega} R(x)dx < \infty$ are assumed. If no a priori information related to $R(x)$ is available, then $R(x) = 1, \forall x \in \Omega$ is used.

The mission space is considered to have $N$ agents. At a given discrete update instant $k$, the position of agent $i$ (i.e., the controllable *local state*) is denoted by $s_{i,k} \in F \subset \mathbb{R}^2$ and the *global state* of the multi-agent system is $\mathbf{s}_k = [s_{1,k}, s_{2,k}, \ldots, s_{N,k}] \in \mathbb{R}^{2N}$. Note that "$\mathbf{s}_k \in \mathbf{F}$" denotes the fact that $s_{i,k} \in F\ \forall i$. In what follows, for notational convenience, the update instant index $k$ is omitted unless it is important.

The sensing capabilities of agent $i$ depend on: (i) a finite *sensing radius* $\delta_i \in \mathbb{R}$ beyond which it cannot detect any events and (ii) the presence of obstacles which hinder its sensing capability. Considering these two factors, a *visibility region* for agent $i$ is defined as $V_i = \{x : \|x - s_i\| \leq \delta_i, \forall \lambda \in (0, 1], (\lambda x + (1 - \lambda)s_i) \in F\}$. Fig. 2·4 is provided to identify all associated geometric parameters in this model.

A *sensing function* $\hat{p}_i(x, s_i)$ is used to quantify the probability that "agent $i$ de-

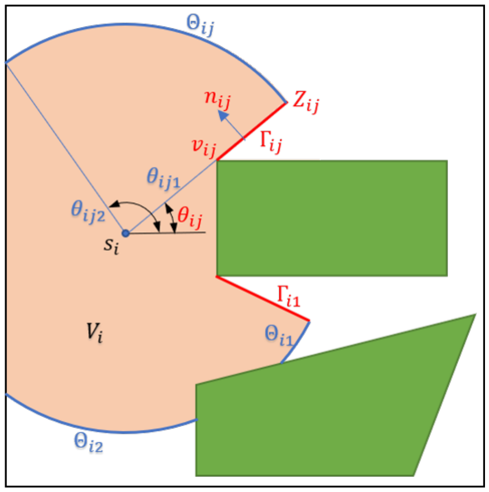


**Figure 2·4:** Mission space with one agent.

tects an event occurring at $x \in F$." Due to the physical limitations mentioned above, $\hat{p}_i(x, s_i) = \mathbf{1}\{x \in V_i\} p_i(x, s_i)$ where $\mathbf{1}\{\cdot\}$ is the usual indicator function and $p_i(x, s_i)$ is defined so that $p_i : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ and is differentiable and monotonically decreasing in $D_i(x) \equiv \|x - s_i\|$. For example, $p_i(x, s_i) = p_{0i} e^{-\lambda_i D_i(x)}$ is a typical choice. However, note that $\hat{p}_i(x, s_i)$ is strictly discontinuous w.r.t. $x$, $s_i$ or $D_i(x)$. Assuming independently detecting agents, the *joint detection probability* $P(x, \mathbf{s})$, i.e., the probability of "detecting an event occurring at $x \in F$ by at least one agent," is given by

$$P(x, \mathbf{s}) = 1 - \prod_{i=1}^{N} [1 - \hat{p}_i(x, s_i)]. \tag{2.37}$$

Combining the event density and joint detection probability, the objective function $H(\mathbf{s})$ of the coverage problem given in (Zhong and Cassandras, 2011) is

$$H(\mathbf{s}) = \int_F R(x) P(x, \mathbf{s}) dx, \tag{2.38}$$

and the multi-agent optimization problem is

$$\mathbf{s}^* = \arg\max_{\mathbf{s} \in \mathbf{F}} H(\mathbf{s}), \tag{2.39}$$

where $\mathbf{s}^*$ represents the optimal agent placement. Note that the objective function in (2.38) is non-linear and non-convex, while the feasible space $\mathbf{F}$ is also non-convex. Therefore, the coverage problem posed in (2.39) has the same structure as the general cooperative multi-agent optimization problem in (2.1). Thus, (2.39) can have multiple locally optimal solutions (even in the simplest configurations). Therefore, the use of the DBS with appropriate boosting functions can aid the agents to escape local optima while solving (2.39).

### 2.3.2 Distributed Optimization Solution

If two agents have an overlap in their visibility regions, they are considered as *neighbors* (Sun et al., 2014). Therefore, the *neighborhood* $B_i$ and the *closed neighborhood* $\bar{B}_i$ of an agent $i$ are the sets defined as $B_i = \{j : V_j \cap V_i \neq \emptyset, i \neq j\}$ and $\bar{B}_i = B_i \cup \{i\}$ respectively. It is assumed that agents share their local state information $s_i$ with their neighbors, so that each agent has knowledge of its *neighborhood state* $\bar{s}_i \equiv \{s_j : j \in \bar{B}_i\}$. An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is used to model inter-agent interactions, where $\mathcal{V} = \{1, 2, \dots, N\}$ and $\mathcal{A} = \{(i, j) : i, j \in \mathcal{V},\ i \neq j,\ j \in B_i\}$.

In (Sun et al., 2014), it is shown that the coverage global objective $H(\mathbf{s})$ in (2.38) can be expressed as $H(\mathbf{s}) = H_i(\bar{s}_i) + H_i^c(s_i^c)$, where

$$H_i(\bar{s}_i) = \int\limits_{V_i} R(x) \prod_{j \in B_i} [1 - \hat{p}_j(x, s_j)]\, p_i(x, s_i) dx, \tag{2.40}$$

and $H_i^c(s_i^c) = \int_F R(x)(1 - \prod_{j \in \mathcal{V}-\{i\}} [1 - \hat{p}_j(x, s_j)])dx$ with $s_i^c = \{s_j : j \in \mathcal{V} - \{i\}\}$. Thus, the $H_i(\bar{s}_i)$ term only depends on the neighborhood state and is called the *local objective* function, while $H_i^c(s_i^c)$ is independent of $s_i$. As a result of this property, the local gradient of agent $i$, defined as $d_i = \frac{\partial H_i(\bar{s}_i)}{\partial s_i} \in \mathbb{R}^2$, is always equal to the global gradient component $\frac{\partial H(\mathbf{s})}{\partial s_i}$. Therefore, each agent $i$ can evaluate its global gradient component using only its own local objective function $H_i(\cdot)$ and the neighborhood state $\bar{s}_i$ and the distributed gradient ascent scheme in (2.3) (i.e., $s_{i,k+1} = s_{i,k} + \beta_{i,k} d_{i,k}$) can be used to solve the problem in (2.39) in a distributed manner.

**Derivation of the Gradient $d_{i,k}$:** Since $d_i \in \mathbb{R}^2$, its components are denoted as $d_i = [d_{iX}, d_{iY}]^T$. Using the Leibniz rule (Flanders, 1973) in (2.38), it can be shown that

$$d_{iX} = \frac{\partial H_i(\bar{s}_i)}{\partial s_{iX}} = \int\limits_{V_i} R(x)\Phi_i(x)\frac{\partial p_i(x,s_i)}{\partial s_{iX}} dx + \int\limits_{\partial V_i} R(x)\Phi_i(x) p_i(x,s_i) \underline{V_x} \cdot \underline{n_x} dl, \quad (2.41)$$

where,

$$\Phi_i(x) = \prod_{j \in B_i} \left[1 - \hat{p}_j(x, s_j)\right]. \quad (2.42)$$

The second term in (2.41) is a line integral over the boundary of the sensing region $\partial V_i$. The terms $\underline{V_x}$ and $\underline{n_x}$ respectively represent the rate of change and the unit normal vector of $\partial V_i$ at $x$ due to an infinitesimal change in $s_{iX}$, where $s_i = [s_{iX}, s_{iY}]^T$.

From Fig. 2·4, notice that the shape of a boundary $\partial V_i$ is formed by: (i) mission space boundaries, (ii) obstacle edges, (iii) obstacle vertices, and, (iv) sensing range. However, when $s_{iX}$ (or $s_{iY}$) is perturbed infinitesimally, $\underline{V_x} \neq 0$ only when $x$ lies on $\partial V_i$ components formed due to the latter two factors. Therefore, the linear segments of $\partial V_i$ formed due to obstacle vertices and circularly shaped curves formed due to finite sensing range are labeled as $\Gamma_i = \{\Gamma_{i1}, \Gamma_{i2}, \ldots\}$ and $\Theta_i = \{\Theta_{i1}, \Theta_{i2}, \ldots\}$, respectively.

The first term in (2.41) can be simplified using the relationship between the $p_i(x, s_i)$ and $D_i(x)$. Further, the behavior of $\underline{V_x} \cdot \underline{n_x}$ on the segments in $\Gamma_i$ and $\Theta_i$ can be used to evaluate the line integral part of (2.41). The resulting $d_{iX}$ expression is

$$\begin{aligned} d_{iX} = \int\limits_{V_i} w_{i1}(x, \bar{s}_i)\frac{(x-s_i)_X}{\|x-s_i\|}dx + \sum_{\Gamma_{ij}\in\Gamma_i} sgn(n_{ijX})\frac{\sin\theta_{ij}}{\|v_{ij}-s_i\|}\int\limits_0^{Z_{ij}} w_{i2}(\rho_{ir}(r), \bar{s}_i)rdr \\ + \sum_{\Theta_{ij}\in\Theta_i} \delta_i \cos\theta \int\limits_{\theta_{ij1}}^{\theta_{ij2}} w_{i3}(\rho_{i\theta}(\theta), \bar{s}_i)d\theta, \end{aligned} \tag{2.43}$$

where, $sgn(\cdot)$ is the signum function and

$$w_{i1}(x, \bar{s}_i) = -R(x)\Phi_i(x)\frac{dp_i(x, s_i)}{dD_i(x)}, \quad w_{i2}(x, \bar{s}_i) = w_{i3}(x, \bar{s}_i) = R(x)\Phi_i(x)p_i(x, s_i),$$

with $\rho_{ir}(r) = \frac{v_{ij}-s_i}{||v_{ij}-s_i||}r + v_{ij}$ and $\rho_{i\theta}(\theta) = s_i + \delta_i[\cos\theta\ \sin\theta]^T$.

As seen in Fig. 2·4, a line segment $\Gamma_{ij} \in \Gamma_i$ is characterized by its: end point $Z_{ij}$, angle $\theta_{ij}$, obstacle vertex $v_{ij}$ and direction $n_{ij} = [n_{ijX}, n_{ijY}]^T$. Thus, each $\Gamma_{ij}$ is a 4-tuple $(Z_{ij}, \theta_{ij}, v_{ij}, n_{ij})$. Similarly, a circular arc segment $\Theta_{ij} \in \Theta_i$ is quantified by starting angle $\theta_{ij1}$ and ending angle $\theta_{ij2}$. Therefore, each $\Theta_{ij}$ term is 2-tuple $(\theta_{ij1}, \theta_{ij2})$.

The complete expression in (2.43) can be thought of as a sum of forces acting on agent $i$, generated by different points $x \in V_i$. In (2.43), the weight function $w_{i1}(x, \bar{s}_i)$ represents the magnitude of the force pulling agent $i$ towards point $x \in V_i$. Similarly, $w_{i2}(x, \bar{s}_i)$ describes the force generated by a point $x \in \Gamma_{ij}$ in the lateral direction to the line $\Gamma_{ij}$ (inwards the region $V_i$). Finally, $w_{i3}(x, \bar{s}_i)$ represents the magnitude of the attraction force generated by (and towards) a point $x \in \Theta_{ij}$. From this interpretation, the gradient component $d_{iX}$ can be viewed as a function of three weight functions: $d_{iX} = d_{iX}(w_{i1}, w_{i2}, w_{i3})$. This representation is instrumental for the construction of boosting functions. The same procedure can be used to derive $d_{iY}$ (and also $K_{1i}$).

### 2.3.3 Designing Boosting Functions

The identified relationship $d_i = d_i(w_{i1}, w_{i2}, w_{i3})$ is now exploited to construct an appropriate expression for the boosted gradient $\hat{d}_{i,k}$ (to be used in (2.5)). Note that each weight function $w_{ij} = w_{ij}(x, \bar{s}_i)$ represents the magnitude component of each of three infinitesimal forces, $j = 1, 2, 3$, acting on agent $i$ generated at a point $x \in V_i$. Note also that $d_{i,k} = 0$ only occurs when all the said infinitesimal forces add up to a resultant force with zero magnitude. Hence, by appropriately transforming the weight functions $\{w_{ij} : j = 1, 2, 3\}$, a valid expression for $\hat{d}_{i,k}$ can be constructed which avoids such equilibrium configurations. Specifically, the weight function transformations

$$\hat{w}_{ij}(x, \bar{s}_i) = \alpha_{ij}(x, \bar{s}_i) w_{ij}(x, \bar{s}_i) + \eta_{ij}(x, \bar{s}_i), \quad j = 1, 2, 3. \tag{2.44}$$

are considered here. Both $\alpha_{ij}, \eta_{ij} : \mathbb{R}^2 \times \mathbb{R}^{2|\bar{B}_i|} \to \mathbb{R}$ are known as *transformation functions*. The resulting boosted gradient $\hat{d}_{i,k}$ expression takes the form

$$\hat{d}_{i,k} = d_{i,k}(\hat{w}_{i1}, \hat{w}_{i2}, \hat{w}_{i3}). \tag{2.45}$$

Compared to heuristic methods where the gradient is randomly perturbed (to escape local optima), the use of boosted gradient $\hat{d}_{i,k}$ in (2.45) is a far more "intelligent" choice as long as each agent $i$ chooses its transformation functions $\alpha_{ij}, \eta_{ij}$, $j = 1, 2, 3$, to trigger a systematic exploration of the mission space. This is discussed next.

**Boosting Function Families:** A *boosting function family* is characterized by the form of the transformation functions $\alpha_{ij}(x, \bar{s}_i)$, $\eta_{ij}(x, \bar{s}_i)$, $j = 1, 2, 3$, in (2.44). As a result, different boosting function families exhibit different properties. A brief review of three boosting function families proposed in (Sun et al., 2014) are given in Appendix B.2.1 as opposed to the two new ones introduced here.

The underlying rationale behind constructing a boosting function family lies in answering the question: "Once an agent converges under the normal gradient-based mode, how can the agent escape the current equilibrium towards a direction giving a high priority to points likely to achieve a higher objective function value?" Towards this goal, to define appropriate $\alpha_{ij}(x, \bar{s}_i)$, $\eta_{ij}(x, \bar{s}_i)$, $j = 1, 2, 3$, in (2.44), the information available to agent $i$ consists of: (i) The neighborhood state $\bar{s}_i$, (ii) The local objective function $H_i(\cdot)$, (iii) The neighboring mission space topological information contained in $\Gamma_i$ and $\Theta_i$ (see Fig. 2·4), (iv) Past state trajectory information $\{s_{i,l} : \ l < k\}$. The three boosting function families proposed in (Sun et al., 2014) use $\bar{s}_i$ and $H_i(\cdot)$, whereas the two new ones make use of $\Gamma_i, \Theta_i$ and $\{s_{i,k} : \ k < k_1\}$ in addition to the information of $\bar{s}_i$ and $H_i(\cdot)$.

For notational convenience, the setting where $\alpha_{ij}(x, \bar{s}_i) = 1$, $\eta_{ij}(x, \bar{s}_i) = 0$, $j = 1, 2, 3$, is referred to as the *default configuration* in (2.44). Also, note that $\kappa$ and $\gamma$ used in defining boosting function families always act as two positive *gain parameters.*

Note that the boosting function families proposed in (Sun et al., 2014) (reported in Appendix B.2.1) are limited to transforming the first integral term of the gradient expression in (2.43), i.e., only the weight $w_{i1}(x, \bar{s}_i)$ is transformed through selecting $\alpha_{i1}(x, \bar{s}_i)$ and $\eta_{i1}(x, \bar{s}_i)$. The two new boosting function families are as follows.

**1. V-Boosting:** The V-Boosting function uses the information of obstacle vertices $v_{ij} \in \Gamma_{ij} \in \Gamma_i$ which lie inside $V_i$ so as to navigate an agent $i$ around surrounding obstacles. This method is inspired by the second integral term in (2.43) which represents the effect of obstacles through $\Gamma_i$ in $V_i$ on agent $i$. Therefore, in V-Boosting, the weight function $w_{i2}(x, \bar{s}_i)$ is transformed via the $\eta_{i2}(x, \bar{s}_i)$ term so that the second integral term in (2.43) is modified. Specifically,

$$\alpha_{i1}(x, \bar{s}_i) = \kappa_1 \Phi_i(x)^{\gamma_1}(1 - p_i(x, s_i)), \tag{2.46}$$

$$\eta_{i2}(x, \bar{s}_i) = \mathbf{1}\{x = Z_{ij}\} \cdot \kappa_2 \|x - s_i\|^{\gamma_2}. \tag{2.47}$$

Moreover, note that $w_{i1}(x, \bar{s}_i)$ is also transformed via the $\alpha_{i1}(x, \bar{s}_i)$ term as in both $\Phi$-Boosting and $P$-Boosting (defined in Appendix B.2.1). In all, the transformation in (2.46) forces agent $i$ to move toward less covered areas while the transformation in (2.47) acts as an attraction force directed towards $Z_{ij} \in \Gamma_{ij}$ (same as in the direction of obstacle vertex $v_{ij}$). The combination of these two influences enables agent $i$ to navigate around obstacles aiming to expand the mission space exploration.

**2. Arc-Boosting:** The Arc-Boosting method uses the information in $\Theta_i$ to transform the weight function $w_{i3}(x, \bar{s}_i)$. This involves the third term in (2.43) which was not previously included in prior work (Zhong and Cassandras, 2011; Sun et al., 2014). Note that $\{\theta_{ij1}, \theta_{ij2}\} = \Theta_{ij} \in \Theta_i$ represents a circular *arc* formed due to the finite sensing range and obstacles. Based on the an agent's location in the mission space relative to the surrounding obstacles, it can have multiple arcs in its boundary $\partial V_i$. For example, the agent in Fig. 2·4 has three such arcs. Under the Arc-Boosting method, first, each arc segment $\Theta_{ij} \in \Theta_i$ is classified into one of three disjoint sets: (i) Attractive Arcs $\Theta_i^+$, (ii) Repulsive Arcs $\Theta_i^-$ and (iii) Neutral Arcs $\Theta_i^0$, based on the metric $A(\Theta_{ij})$:

$$A(\Theta_{ij}) = \frac{1}{(\theta_{ij2} - \theta_{ij1})} \int_{\theta_{ij1}}^{\theta_{ij2}} (1 - \prod_{k \in \bar{B}_i} (1 - \hat{p}_k(\rho_{i\theta}(\theta), s_k)))d\theta,$$

which measures the mean coverage level on the arc segment $\Theta_{ij}$ by the agents in the closed neighborhood $\bar{B}_i$. Specifically, the arc with the maximum $A(\Theta_{ij})$ value is assigned to be a repulsive arc (i.e., in the set $\Theta_i^+$), while the arc with the minimum $A(\Theta_{ij})$ value is assigned to be an attractive arc (i.e., in the set $\Theta_i^-$). The remaining arcs are labeled as neutral (i.e., in the set $\Theta_i^0$). However, it is possible that an equilibrium occurs (i.e., $A(\Theta_{ij})$ are identical for all $j$), which may happen when $B_i = \emptyset$. In this case, a recent state $s_{i,k-K}$, where $K \geq 1$ is used as a parameter of the Arc-Boosting method, selected from the agent's own past state trajectory. Specifically, the arc which is in the direction of $s_{i,k-K}$ (from point $s_i$) is regarded as a repulsive arc while all other arcs are labeled as attractive. Based on the arc partition given by $\Theta_i^+, \Theta_i^-$ and $\Theta_i^0$, the Arc-Boosting function family is formally defined by the weight function $w_{i3}(x, \bar{s}_i)$ transformation given by

$$\alpha_{i3}(x, \bar{s}_i) = \mathbf{1}\{\Theta_{ij} \in \Theta_i^0\}, \tag{2.48}$$

$$\eta_{i3}(x, \bar{s}_i) = [\mathbf{1}\{\Theta_{ij} \in \Theta_i^+\} - \mathbf{1}\{\Theta_{ij} \in \Theta_i^-\}] \cdot F_c(\kappa, \gamma). \tag{2.49}$$

In (2.49), the value of the term in brackets is either $1, -1$ or $0$ depending on whether $\Theta_{ij}$ belongs to $\Theta_i^+, \Theta_i^-$ or $\Theta_i^0$ respectively. The term $F_c(\kappa, \gamma)$ is a gain factor where a typical choice would be of the form $F_c(\kappa, \gamma) = \kappa e^{\gamma}$.

The motivation behind the Arc-Boosting method is to encourage agent $i$ to: (i) Move away from highly covered regions (i.e., from repulsive arcs), (ii) Move towards less covered regions (i.e., towards attractive arcs), and, (iii) Move continuously towards unexplored regions (i.e., towards an opposing direction to the already visited point $s_{i,k-K}$). As will be seen in Section 2.3.5, the Arc-Boosting family has been found to be the most effective in handling the presence of multiple obstacles in $V_i$.

### 2.3.4 DBS for Coverage

The final step is to implement the DBS in Fig. 2·2 for the class of coverage problems (complete implementation details are given in (Welikala and Cassandras, 2019b)). Convergence is guaranteed through Theorem 2.2 by checking that Assumptions 2.1-2.3 and 2.5 are satisfied. Assumption 2.1 holds for the coverage problem due to two reasons: (i) The Lipshitz constant $K_{1i}$ of $\nabla H_i(\bar{s}_i)$ can be locally evaluated and will always be finite as shown in (Welikala and Cassandras, 2019b). (ii) $H_{UB} = \int_{V_i} R(x)dx$ is a typical upper bound for $H_i(\bar{s}_i)$ as $\int_{\Omega} R(x)dx < \infty$ is already enforced in the coverage problem formulation. Assumption 2.2 holds for the coverage problem because information sharing capability is already assumed in the basic coverage problem framework (Sun et al., 2014). However, the following lemma is useful in asserting that no additional communication bandwidth is required to satisfy this assumption.

**Lemma 2.7.** *For the class of coverage problems, any agent $i \in \mathcal{V}$ can locally compute $d_{ij} = \frac{\partial H_j(\bar{s}_j)}{\partial s_i}$ value $\forall j \in \bar{B}_i$.*

*Proof.* See Appendix A.2.6. □

Assumption 2.3 has been previously justified for the general setting in Section 2.2 using Lemmas 2.3 and 2.5. However, to ensure this assumption is satisfied in coverage problems, the parameter $T_i$ was observed during all simulations presented in Section 2.3.5 for all agents. In all occasions, $T_i$ was found to be a finite consistent with Assumption 2.3. One such observed $T_i$ value distribution is given in Fig. 2·5, where 99.1% of the time $T_i \leq 10$. Finally, as pointed out in Section 2.2, Assumption 2.5 is trivial and will hold for any general problem including coverage problems.

### 2.3.5 Simulation Results

For the class of coverage problems, the proposed DBS (with all boosting function families) and the methods proposed in (Zhong and Cassandras, 2011; Sun et al., 2014)

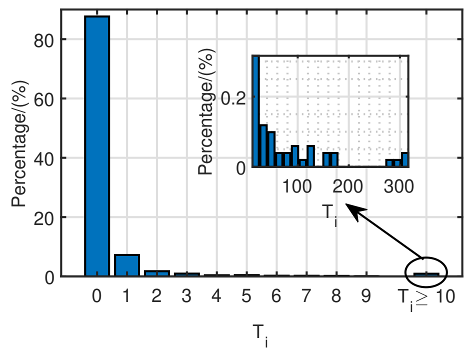


**Figure 2·5:** Percentage occurrence of different $T_i$ values (In Assumption 2.3 for the simulation which produced the result in Fig. 2·6f).

were implemented in an interactive JavaScript-based simulator which is available at `http://www.bu.edu/codes/simulations/shiran27/CoverageFinal/` and may be used by the reader to reproduce the reported results. The boosting function parameters used in generating the reported results (i.e., $\kappa, \gamma$) are given in Table 2.1.

Based on the obstacle arrangement, four different mission space configurations named 'General','Room','Maze', and 'Narrow' are considered in the simulations. In Figs. 2·6, 2·8, and, 2·7 obstacles are shown as green-colored blocks and agent locations are shown in red-colored dots. In all experiments, agents have been initialized at the top left corner of the mission space. Further, light green-colored areas indicate higher coverage levels while yellow-colored areas indicate the opposite.

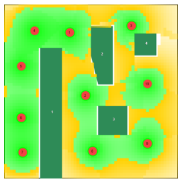
As the first step, a set of experiments was conducted with $N = 10$ agents and three different algorithms were tested: (i) The conventional distributed gradient ascent method proposed in (Zhong and Cassandras, 2011) (labelled "GA"), (ii) The CBS proposed in (Sun et al., 2014), and, (iii) The DBS proposed in this thesis. Results obtained from the GA method are shown in Figs. 2·6a, 2·6c, 2·6e, and, 2·6g. The corresponding objective function values are listed in Table 2.2 under the column: 'Reference Level $H(\mathbf{s}^1)$'. Similarly, results obtained from the CBS and DBS methods (under different boosting function families) are listed in the remaining columns of Table 2.2 - as the increment achieved in the coverage objective value with respect to the reference level $H(\mathbf{s}^1)$. The cases with the highest coverage objective value increments are shown in bold letters and they are illustrated in Figs. 2·6b, 2·6d, 2·6f and 2·6h. The results in Table 2.2 show that the distributed Arc-Boosting (labeled "AB") and

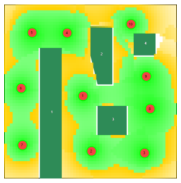
**Table 2.1:** Boosting function parameters used in simulations.

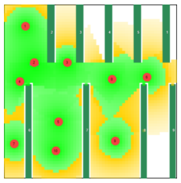
| **Boosting Method** | **Associated Default Parameters** |
|---|---|
| $P$-Boosting | $\kappa = 1,\ \gamma = 1$ |
| Neighbor-Boosting | $\kappa = 10000,\ \gamma = 1$ |
| $\Phi$-Boosting | $\kappa = 4,\ \gamma = 2$ |
| V-Boosting | $\kappa_1 = 10,\ \kappa_2 = 5,\ \gamma_1 = 1$, and, $\gamma_2 = 1$ |
| Arc-Boosting | $\kappa = 1,\ \gamma = 1,\ K = 50,\ T_D = 5$ |

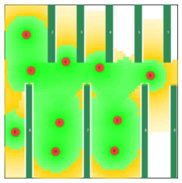
**Table 2.2:** Coverage objective value **increment** (+/-) achieved by different boosting schemes (See Fig. 2·6).

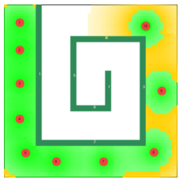
| | | **Reference Level $H(\mathbf{s}^1)$** | **Coverage objective value increment occurred with respect to the 'Reference Level $H(\mathbf{s}^1)$'** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Configuration | | Gradient | $P$-Boosting | | Neighbor Boo. | | Φ-Boosting (ΦB) | | V-Boosting (VB) | | Arc-Boosting (AB) | |
| Obstacles | N | Ascent (GA) | Centr. | Decen. | Centr. | Decen. | Centr. | Decen. | Centr. | Decen. | Centr. | Decen. |
| General | 10 | **158,821** | +235 | +3684 | +235 | +3676 | +243 | +3674 | +2453 | +3621 | +3553 | **+3739** |
| Room | 10 | **143,583** | +1578 | +2680 | +2374 | +968 | +1578 | +2626 | +1739 | +2455 | +1578 | **+2768** |
| Maze | 10 | **120,343** | +25937 | +25897 | +19443 | +25895 | +26952 | +23868 | +19970 | +25702 | +25945 | **+27142** |
| Narrow | 10 | **169,793** | +9204 | +8835 | +15258 | +9391 | +15008 | +9376 | +14969 | **+15286** | +15238 | +15120 |

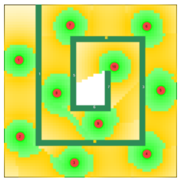

**(a)** GA: 158, 821

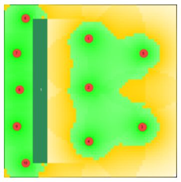

**(b)** +2.354% AB: 162, 560


**(c)** GA: 143, 583


**(d)** +1.928% AB: 146, 351


**(e)** GA: 120, 343


**(f)** +22.55% AB: 147, 485


**(g)** GA: 169, 793

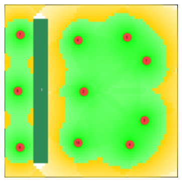

**(h)** +9.003% VB: 185, 079

**Figure 2·6:** Maximum coverage improvement achieved due to boosting for $N = 10$ (See Tab. 2.2).

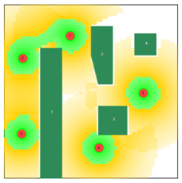
distributed V-Boosting (labeled "VB") schemes outperform all other methods for all tested obstacle configurations when $N = 10$.

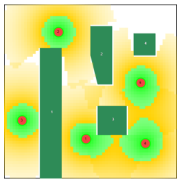
Moreover, to further investigate the performance of the distributed V-Boosting and Arc-Boosting methods, simulation results were generated with moderate $N$ values such as $N = 5, 6$. The corresponding results are shown in Table 2.3 and Fig. 2·7. Finally, extreme situations (i.e., more prone to local optima) where very few agents are deployed (i.e., $N = 1, 2$) were also investigated and simulation results obtained are shown in Table 2.4 and Fig. 2·8. Note that these additional experimental results also highlight the impact of the distributed Arc-Boosting and V-Boosting schemes.

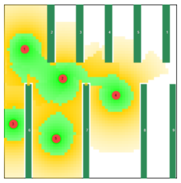
In summary, these simulation results show that the boosting function approach

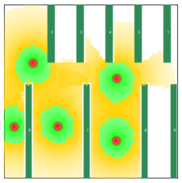
**Table 2.3:** Coverage objective value for cases with $N = 5, 6$ with decentralized boosting (See Fig. 2·7).

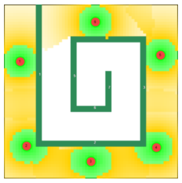
| Configuration | | Gradient | Decentralized | Decentralized |
|---|---|---|---|---|
| Obstacles | N | Ascent (GA) | V-Boosting | Arc-Boosting |
| General | 5 | **93,637** | **97,214** | 96,832 |
| Maze | 6 | **90,953** | 94,026 | **94,436** |
| Room | 5 | **86,638** | 89,078 | **89,088** |
| Narrow | 6 | **101,976** | 116,481 | **129,476** |

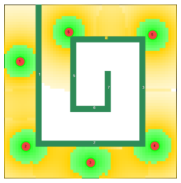

**(a)**
GA: $93,637$

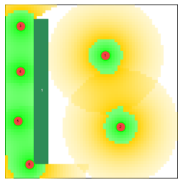

**(b)** +3.820%
VB: $97,214$


**(c)**
GA: $86,638$


**(d)** +2.828%
AB: $89,088$


**(e)**
GA: $90,953$


**(f)** +3.829%
AB: $94,436$


**(g)**
GA: $101,976$

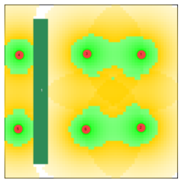

**(h)** +26.97%
AB: $129,476$

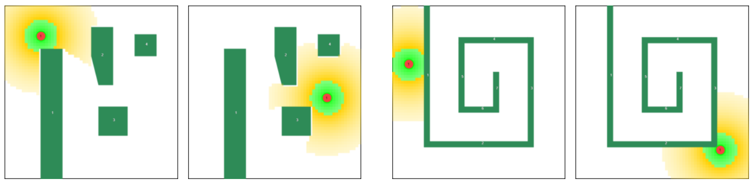
**Figure 2·7:** Maximum coverage improvement achieved due to boosting for $N = 5, 6$ (See Tab. 2.3).

can successfully escape local optima which limits the conventional gradient ascent based method. Further, the systematic gradient transformation process achieved by the specifically designed boosting function families, along with the introduced DBS, delivers superior objective function values compared to conventional gradient ascent based methods.

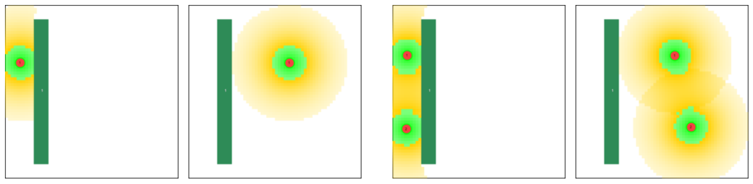
Note that whenever the DBS was used, the variable step size method involved in Theorem 2.2 was used to guarantee convergence. As an example, Fig. 2·9 illustrates the observed step size sequence and the associated gradient sequence of a typical agent ($i = 4$) during the simulation which leads to the result shown in 2·6h. Moreover, Table 2.5 provides a comparison of coverage objective and convergence time values observed for the DBS when fixed and variable step sizes are used. Note that the use of variable

**Table 2.4:** Coverage objective value for cases with $N = 1, 2$ with decentralized boosting (See Fig. 2·8).

| Configuration Obstacles | N | Gradient Ascent (GA) | Decentralized V-Boosting | Decentralized Arc-Boosting |
|---|---|---|---|---|
| General | 1 | **20,494** | 20,404 | **23,193** |
| Maze | 1 | **14,759** | 14,774 | **17,090** |
| Narrow | 1 | **13,669** | **30,259** | 30,178 |
| Narrow | 2 | **26,258** | **58,693** | 58,681 |



**(a)** GA: $20,494$ **(b)** $+13.17\%$ AB: $23,193$ **(c)** GA: $14,759$ **(d)** $+15.79\%$ AB: $17,090$



**(e)** GA: $13,669$ **(f)** $+121.4\%$ VB: $30,259$ **(g)** GA: $26,258$ **(h)** $+123.5\%$ VB: $58,693$

**Figure 2·8:** Maximum coverage improvement achieved due to boosting for $N = 1, 2$ (See Tab. 2.4).

**Table 2.5:** Comparison of coverage objective and convergence time values observed for the DBS with fixed and variable steps.

| Boosting Method | $(N = 8)$ Configuration | $H(s^*)$ Fixed steps | $H(s^*)$ Variable steps | Convergence Time Fixed steps | Convergence Time Variable steps |
|---|---|---|---|---|---|
| V-Boosting | General | 140,592 | 140,649 | 550.7 | 91.3 |
| | Room | 127,557 | 127,517 | 613.5 | 140.3 |
| | Maze | 120,832 | 121,231 | 302.2 | 134.1 |
| | Narrow | 163,478 | 155,528 | 415.7 | 161.8 |
| Arc-Boosting | General | 140,615 | 140,542 | 80.3 | 104.9 |
| | Room | 127,647 | 127,455 | 390.0 | 158.1 |
| | Maze | 119,967 | 121,231 | 151.7 | 125.0 |
| | Narrow | 155,641 | 155,485 | 127.3 | 88.1 |
| Average: | | 137,041 | 136,205 | 328.9 | 125.4 |

step sizes has improved (i.e., reduced) the convergence time by 61.9% (i.e., by 203.5$s$). These convergence times were observed on an Intel® Core™ i7-8700 CPU @3.20 GHz Processor with a 32 GB RAM.

To conclude this section, the effects of decentralization is addressed as these sim-

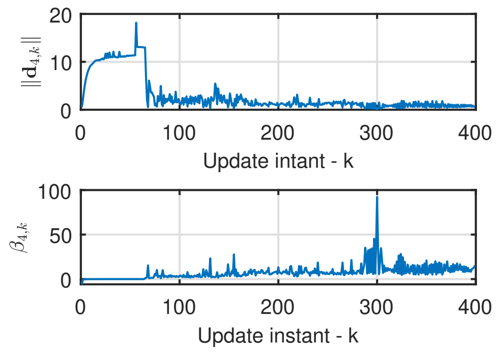


**Figure 2·9:** Variation of gradient magnitude and the step size for agent $i = 4$ during the simulation which yielded Fig. 2·6h.

ulation results show the DBS outperforming the CBS in every aspect. When all simulations carried out for $N = 10$ were considered (given in table 2.2), on average (per simulation), the convergence time to the final optimal solution was improved (i.e., reduced) by 39.97% (approximately 165.2 $s$) due to the distributed nature of the DBS relative to a centralized approach. Further, due to decentralization, on average (per simulation), the final coverage cost achieved was increased by 0.381% (approximately 451 units). Finally, decentralization has the inherent advantages of reducing communication and implementation costs compared to a centralized solution.

## 2.4 Summary

This chapter of the thesis proposed a boosting functions approach to overcome the issue of multiple local optima arising in cooperative multi-agent optimization problems with non-convex objective functions. First, a distributed boosting scheme was proposed together with an optimal variable step size selection mechanism to guarantee its convergence. Next, providing more intuition about the boosting function design process, for a class of multi-agent coverage problems (Zhong and Cassandras, 2011), two new boosting function families named *Arc-Boosting* and *V-Boosting* were designed. Finally, the same class of multi-agent coverage problems was used to illustrate the effectiveness of the proposed DBS, variable step size selection technique and boosting function families (i.e., of the proposed boosting function approach). Numerical results show that the proposed boosting functions approach can successfully escape local optima that limits the conventional gradient ascent based methods and achieve superior performance levels without significantly affecting the involved computational cost - when addressing cooperative multi-agent optimization problems.

# Chapter 3

# Greedy Initialization for Multi-Agent Optimization

This chapter focuses on exploring how greedy initialization introduced in Section 1.2.2 can contribute to overcome the issue of converging to poor local optima faced in co-operative multi-agent optimization problems with non-convex objective functions. In particular, the objective function's submodularity property is exploited to establish performance bounds for the obtained solutions. In situations where the objective function is non-convex, having such a performance bound is highly valued as it indicates the closeness of the obtained solution to the global optimal. Therefore, this chapter focuses explicitly on submodular function optimization, performance bounds and their application to the class of multi-agent coverage problems (Zhong and Cassandras, 2011).

The sections of this chapter are as follows. First, Section 3.1 briefly revisits the required preliminary concepts. Second, Section 3.2 reviews all the curvature concepts available in the literature for the class of submodular maximization problems. Next, Section 3.3 introduces two new curvature concepts that can be effective compared to the existing curvature concepts in providing improved performance bounds. Then, Section 3.4 models the class of multi-agent coverage problems as a class of submodular maximization problems and establishes several of its underlying structural properties. Also, in the same section, numerical results are provided to highlight the contributions. Finally, some concluding remarks are provided in Section 3.5.

## 3.1 Preliminary Concepts

Consider a finite *ground set* denoted as $X = \{x_1, x_2, \ldots, x_n\}$ that represents all the possible actions/options available for an agent in a multi-agent optimization problem (analogues to a discretized version of the feasible space $F$ in (2.38) and (2.39)). The *set-function* $f : 2^X \to \mathbb{R}$ is considered as the objective function (also called the *set-objective*). Note that $2^X$ is the *power-set* of $X$ (i.e., the set of all subsets of $X$).

### 3.1.1 Basic Set-Function Properties

Some formal definitions of basic set-function properties used here are as follows.

**Definition 3.1.** *The discrete derivative (also called the* ***marginal gain****) function of the set-function $f$ at $A \subset X$ in the direction of element $x \in X \backslash A$ is defined as,*

$$\Delta f(x|A) \triangleq f(A \cup \{x\}) - f(A).$$

*This notation is also used more liberally as $\Delta f(B|A) \triangleq f(A \cup B) - f(A)$, for $A, B \subseteq X$.*

**Definition 3.2.** *Each of the following statements is equivalent and implies the* ***submodularity*** *of the set function $f$.*

1. $f(A \cup B) + f(A \cap B) \leq f(A) + f(B), \quad \forall A, B \subseteq X,$
2. $\Delta f(x|A) \leq f(x|B)$*, $\forall x, B, A$, where $B \subseteq A \subseteq X$ and $x \in X \backslash A$,*
3. $\Delta f(x_i|A \cup \{x_j\}) \leq \Delta f(x_i|A)$*, $\forall A, x_i, x_j$, where $A \subseteq X$ and $x_i, x_j \in X \backslash A$ with $x_i \neq x_j$.*

Note that "$\cdot\backslash\cdot$" stands for the set subtraction operation and the second equivalent form given above is commonly known as the "Diminishing Returns" property.

**Definition 3.3.** *The set-function $f$ is: (i)* ***supermodular*** *if its negation $(-f)$ is submodular, and, (ii)* ***modular*** *if it is both submodular and supermodular.*

**Definition 3.4.** *The set-function $f$ is: (i)* ***monotone*** *if $f(B) \leq f(A)$, $\forall B, A$, where $B \subseteq A \subseteq X$, and, (ii)* ***normalized*** *if $f(\emptyset) = 0$.*

**Definition 3.5.** *The set-function $f$ is said to be a* ***polymatroid*** *function if it is submodular, monotone and normalized.*

### 3.1.2 Matroid Constraints

Depending on the application, the set-objective function $f$ can have a constrained domain smaller than the power set $2^X$. This can be a result of either (i) $f$ being not defined on the entire power set $2^X$, or, (ii) when evaluating $f$ on all possible sets in $2^X$ is not desirable. Such a situation is modeled by introducing *set-variable constraints* for the set-objective $f(Y)$. Formally, the concept of "matroids" is used to characterize different forms of common set-variable constraints.

Let $\mathcal{I}$ be a non-empty collection of subsets of $X$ (i.e. $\mathcal{I} \subseteq 2^X$). Then, a pair $\mathcal{M} = (X, \mathcal{I})$ can have different properties as given below.

**Definition 3.6.** *A pair $\mathcal{M} = (X, \mathcal{I})$ is **hereditary** if $\forall A, B$ such that $A \in \mathcal{I}$ and $B \subseteq A \implies B \in \mathcal{I}$ (this is same as calling $\mathcal{I}$ an **independent system**).*

**Definition 3.7.** *A pair $\mathcal{M} = (X, \mathcal{I})$ follows the **augmentation** property if for all $A, B \in \mathcal{I}$ with $|B| \leq |A|$, $\exists x \in A \backslash B$ such that $B \cup \{x\} \in \mathcal{I}$.*

**Definition 3.8.** *A pair $\mathcal{M} = (X, \mathcal{I})$ is called a **matroid** if it follows both the **hereditary** and **augmentation** properties. Also, the **rank** of a such matroid is the cardinality of its largest set in $\mathcal{I}$.*

**Definition 3.9.** *A matroid $\mathcal{M} = (X, \mathcal{I})$ is called a **uniform matroid** of rank $N$ if,*

$$\mathcal{I} = \{Y : Y \subseteq X, |Y| \leq N\}.$$

*This is also known as a **cardinality constraint** of rank $N$ (denoted by $\mathcal{I}_N \triangleq \mathcal{I}$).*

### 3.1.3 General Submodular Maximization

Given a ground set $X$, a pair $\mathcal{M} = (X, \mathcal{I})$ and a set-objective function $f : 2^X \to \mathbb{R}$, the problem of finding the set $Y \in \mathcal{I}$ that maximizes $f$ is a widely studied problem in the field of *combinatorial optimization.* This problem is formally stated as

$$Y^* = \underset{Y \in \mathcal{I}}{\arg\max}\ f(Y), \tag{3.1}$$

and it is NP-hard (Liu et al., 2019). However, one trivial approach to solve (3.1) is to use a *brute force* search algorithm which evaluates $f$ over each element in $\mathcal{I}$. Even though such an approach can give the exact global optimal $Y^*$, in many applications of interest, it is computationally intractable due to the involved search space size $|\mathcal{I}|$.

As a remedy, *greedy algorithms* are widely used to generate a reasonable approximate (sub-optimal) solution to (3.1). Such a solution is generally referred to as a *greedy solution* and denoted by $Y^G$. A typical greedy algorithm is given in Alg. 3.1.

**Algorithm 3.1** Centralized greedy algorithm to solve (3.1)

```
1: $Z = \emptyset$;
2: while True do
3:     $z = \arg\max_{\{x:Z\cup\{x\}\in\mathcal{I}\}} \Delta f(x|Z)$;
4:     if $\Delta f(z|Z) > 0$ then
5:         $Z = Z \cup \{z\}$;
6:     else
7:         Break;
8:     end if
9: end while
10: $Y^G := Z$; Return $Y^G$;
```

### 3.1.4 Performance Bound Guarantees

Even though a greedy solution of (3.1) is often sub-optimal (i.e., $f(Y^G) \leq f(Y^*)$), the following concepts are used to characterize the closeness between $f(Y^G)$ and $f(Y^*)$.

**Definition 3.10.** *The* ***performance ratio*** $L^{PR}$ *of a greedy solution* $Y^G$ *obtained for a problem of the form* (3.1) *is defined as* $L^{PR} \triangleq \frac{f(Y^G)}{f(Y^*)}$. *A corresponding* ***performance bound*** *is a theoretically established lower bound* $\beta$ *for* $L^{PR}$. *Therefore,*

$$\beta \leq L^{PR} \triangleq \frac{f(Y^G)}{f(Y^*)}.$$

Note that if $\beta$ is closer to 1, it implies that the performance of the greedy solution is closer to that of the global optimal solution ($f(Y^G) \simeq f(Y^*)$). Hence, a performance bound $\beta$ conveys the effectiveness of using a greedy method for a given problem. The most *fundamental performance bounds* established in the seminal paper (Nemhauser et al., 1978) for a problem of the form (3.1) are compiled in the following theorem.

**Theorem 3.1.** *(Nemhauser et al., 1978) For the set-function maximization problem* (3.1)*, when the greedy algorithm in Alg. 3.1 is used, if the set-objective function is polymatroid, the following performance bounds (denoted by $\beta_f = \beta$) can be established:*

- $\beta_f = \frac{1}{2}$*, if the pair $\mathcal{M} = (X, \mathcal{I})$ is hereditary (i.e., $\mathcal{I}$ is an independent system).*
- $\beta_f = 1 - (1 - \frac{1}{N})^N$*, if the pair $\mathcal{M} = (X, \mathcal{I})$ is a uniform matroid of rank $N$.*
- $\beta_f = 1 - \frac{1}{e}$ *with uniform matroid constraints.*

## 3.2 A Review of Different Curvature Concepts

Structural properties of the set-objective function $f$, the ground set $X$ and the constraints $Y \in \mathcal{I}$ involved in the combinatorial optimization problem (3.1) can be exploited to establish tighter (i.e., closer to 1 compared to $\beta_f$) performance bounds for the same greedy solution $Y^G$ given by Alg. 3.1. Note that a such tighter performance bound is preferred (over $\beta_f$) because it allows us to: (i) have a more accurate sense of proximity of the greedy solution ($Y^G$) to the optimality ($Y^*$) and (ii) make more informed decisions regarding spending extra resources to seek a better solution. In particular, different "curvature" measures that characterize such structural properties are often used to obtain improved/tighter performance bounds. In this section, four established curvature measures along with their respective performance bounds are briefly reviewed, outlining their properties, strengths and weaknesses. Note also that the following assumption is made regarding problem (3.1) unless stated otherwise.

**Assumption 3.1.** *In the main problem* (3.1)*, the set-objective function $f$ is a polymatroid and the pair $M = (X, \mathcal{I})$ is a uniform matroid of rank $N$ (i.e., $\mathcal{I} = \mathcal{I}_N$).*

### 3.2.1 Total Curvature (Conforti and Cornuéjols, 1984)

For the problem in (3.1), the *total curvature* measure $\alpha_t$ is defined as

$$\alpha_t \triangleq \alpha_t(f, 2^X) = \max_{\substack{x:x\in X,\\ 0<\Delta f(x|\emptyset)}} \left[1 - \frac{\Delta f(x|X\backslash\{x\})}{\Delta f(x|\emptyset)}\right]. \tag{3.2}$$

The corresponding performance bound $\beta_t$ is given by

$$\beta_t \triangleq \frac{1}{\alpha_t}\left[1 - \left(1 - \frac{\alpha_t}{N}\right)^N\right] \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.3}$$

Note that $0 \leq \alpha_t \leq 1$ and $\beta_t$ is a decreasing function with respect to $\alpha_t$. Therefore, lower total curvature measures ($\alpha_t$ closer to 0) deliver better performance bounds ($\beta_t$ closer to 1). In contrast, if $\alpha_t$ is closer to 1, the corresponding performance bound $\beta_t$ reveals the fundamental bound $\beta_f$ given in Theorem 3.1.

**Remarks:** The $\alpha_t$ in (3.2) can be simplified (assuming $\forall x \in X, 0 < \Delta f(x|\emptyset)$) into

$$\alpha_t = 1 - \min_{x\in X}\left[\frac{\Delta f(x|X\backslash\{x\})}{\Delta f(x|\emptyset)}\right].$$

From submodularity of $f$, $\Delta f(x|X\backslash x) \leq f(x|\emptyset), \forall x \in X$. Therefore, improved performance bounds can be obtained if $f$ and $X$ in (3.1) are such that $\Delta f(x|\emptyset) \simeq \Delta f(x|X\backslash x), \forall x \in X$. Moreover, as $\Delta f(x|X\backslash x) = f(X) - f(X\backslash x)$, evaluating $\alpha_t$ requires evaluating $f(X)$, which might be ill-defined. For example, consider a situation where the domain of $f$ is strictly constrained to $\mathcal{I}_N$ (i.e., when $f : \mathcal{I}_N \rightarrow \mathbb{R}$).

### 3.2.2 Greedy Curvature (Conforti and Cornuéjols, 1984)

The *greedy curvature* measure $\alpha_g$ is computed based on successive solution-sets that the greedy algorithm generates (i.e., in Alg. 3.1) when solving (3.1). These solution-sets are respectively denoted by $\emptyset = Z^0 \subseteq Z^1 \subseteq Z^2 \subseteq \cdots \subseteq Z^N$, where $Z^N$ is the final greedy solution (i.e., $Y^G = Z^N$). Specifically, $\alpha_g$ is defined as

$$\alpha_g \triangleq \max_{0 \leq i \leq N-1} \left[ \max_{x \in X^i} \left( 1 - \frac{\Delta f(x|Z^i)}{\Delta f(x|\emptyset)} \right) \right], \tag{3.4}$$

where $X^i = \{x : x \in X \backslash Z^i, (Z^i \cup \{x\}) \in \mathcal{I}, \Delta f(x|\emptyset) > 0\}$ (i.e., the set of feasible options in the $(i+1)^{\text{th}}$ iteration of the greedy algorithm). The corresponding performance bound $\beta_g$ is given by

$$\beta_g \triangleq 1 - \alpha_g(1 - \frac{1}{N}) \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.5}$$

Note that $0 \leq \alpha_g \leq 1$ and $\beta_g$ is a decreasing function in $\alpha_g$. Therefore, when $\alpha_g \to 0$, $\beta_g \to 1$. However, when $\alpha_g \to 1$, unlike in the case of $\beta_t$, $\beta_g \to \frac{1}{N} < \beta_f$.

**Remarks:** The expression of $\alpha_g$ in (3.4) can be written as

$$\alpha_g = 1 - \min_{0 \leq i \leq N-1} \left[ \min_{x \in X^i} \left( \frac{\Delta f(x|Z^i)}{\Delta f(x|\emptyset)} \right) \right].$$

From submodularity of $f$, $\Delta f(x|Z^i) \leq \Delta f(x|\emptyset)$. Therefore, to get tighter performance bounds, $f, X$ and $Z^i, i \in \{0, 1, \ldots, N\}$ of problem (3.1) should be such that $\Delta f(x|Z^i) \simeq \Delta f(x|\emptyset), \forall x \in X \backslash Z^i, i \in \{0, 1, \ldots, N-1\}$. Moreover, note that $\beta_g$ in (3.5) can be computed in parallel to executing the greedy algorithm without requiring any additional numerical evaluations of $f$. Note also that unlike $\beta_t$ in (3.3), $\beta_g$ in (3.5) can be computed even when the domain of $f$ is constrained to $\mathcal{I}_N$.

### 3.2.3 Elemental Curvature (Wang et al., 2016)

For the problem in (3.1), the *elemental curvature* measure $\alpha_e$ is defined as

$$\alpha_e \triangleq \max_{\substack{(Y, x_i, x_j): Y \subset X, \\ x_i, x_j \in X \backslash Y,\ x_i \neq x_j.}} \left[ \frac{\Delta f(x_i | Y \cup \{x_j\})}{\Delta f(x_i | Y)} \right]. \tag{3.6}$$

The corresponding performance bound $\beta_e$ is given by

$$\beta_e \triangleq 1 - \left(\frac{\alpha_e + \alpha_e^2 + \cdots + \alpha_e^{N-1}}{1 + \alpha_e + \alpha_e^2 + \cdots + \alpha_e^{N-1}}\right)^N \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.7}$$

Note that $0 \leq \alpha_e \leq 1$ and $\beta_e$ is a decreasing function with respect to $\alpha_e$. Therefore, when $\alpha_e \to 0$, $\beta_e \to 1$. Also, when $\alpha_e \to 1$, similar to the case of $\beta_t$, $\beta_e \to \beta_f$.

**Remarks:** According to the third condition in Def. 3.2, submodularity of $f$ directly depends on the condition $\Delta f(x_i|Y \cup \{x_j\}) \leq \Delta f(x_i|Y)$ for all feasible $(Y, x_i, x_j)$ choices. However, if $\Delta f(x_i|Y \cup \{x_j\}) = \Delta f(x_i|Y)$ occurs for some feasible combination of $(Y, x_i, x_j)$, it means that the set-objective function $f$ is *modular* (see Def. 3.3) in that specific region. According to (3.6), such an existence of a modular region of $f$ over $X$ causes $\alpha_e = 1$ resulting $\beta_e = \beta_f$. A trivial situation where this ($\beta_e = \beta_f$) occurs is when $f, X$ in problem (3.1) is such that $\exists x_i, x_j \in X$ with $x_i \neq x_j$ where $f(\{x_i\}) + f(\{x_j\}) = f(\{x_i, x_j\})$. Therefore, it is clear that the elemental curvature based performance bound $\beta_e$ fails (i.e., $\beta_e = \beta_f$) unless $f$ is *strictly submodular* everywhere over its domain, i.e., $\Delta f(x_i|Y \cup \{x_j\}) \ll \Delta f(x_i|Y)$ for all feasible $(Y, x_i, x_j)$ choices.

### 3.2.4 Partial Curvature (Liu et al., 2018)

The *partial curvature* measure $\alpha_p$ has been introduced as an alternative to the total curvature measure $\alpha_t$ in (3.2). This can be used when the set objective function $f$ has a strictly constrained domain, i.e., when $f : \mathcal{I} \to \mathbb{R}$ with $\mathcal{I} \subset 2^X$ (where $\alpha_t$ will be ill-defined due to the involved $f(X)$ term in (3.2)). In particular, $\alpha_p$ is defined as

$$\alpha_p = \alpha_p(f, \mathcal{I}) = \max_{\substack{(A,x):x\in A\in\mathcal{I}\\ \Delta f(x|\emptyset)>0}} \left[1 - \frac{\Delta f(x|A\backslash\{x\})}{\Delta f(x|\emptyset)}\right]. \tag{3.8}$$

The corresponding performance bound $\beta_p$ is given by the following theorem.

**Theorem 3.2.** *(Liu et al., 2019) Let $f : \mathcal{I}_N \to \mathbb{R}$ be a polymatroid function.* ***If*** *there exists a set-function $g : 2^X \to \mathbb{R}$ which is: (i) an extension of $f$, (ii) a polymatroid* *function, and (iii) satisfies the binding condition* $\alpha_p(f,\mathcal{I}_N)=\alpha_t(g,2^X)$*,* ***then****, for the problem in* (3.1)*, the greedy solution has a performance bound* $\beta_p$ *where*

$$\beta_p \triangleq \frac{1}{\alpha_p}\left(1-\left(1-\frac{\alpha_p}{N}\right)^N\right) \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.9}$$

A brief summary of the findings in (Liu et al., 2018) and explanations of the extra conditions mentioned in Theorem 3.2 are provided in Appendix B.3.1. However, note that $\beta_p$ in (3.9) is independent of the function $g$ mentioned in Theorem 3.2 (of which the existence needs to be proven prior to using $\beta_p$). Moreover, note that $\beta_p$ in (3.9) and $\beta_t$ in (3.3) has identical forms - enabling a direct comparison between $\alpha_t$ and $\alpha_p$. The work in (Liu et al., 2018) (see also Appendix B.3.1) has shown that $\alpha_p(f,\mathcal{I}_N)\leq\alpha_t(f,2^X)$, which implies that $\beta_p\geq\beta_t$, i.e., $\beta_p$ is tighter than $\beta_t(\geq\beta_f)$.

**Remarks:** Note that evaluating $\alpha_p$ in (3.8) is much difficult compared to evaluating $\alpha_t$ in (3.2) as $\alpha_p$ involves an optimization over a set-variable. However, using $B=A\backslash\{x\}$ and $\mathcal{I}=\mathcal{I}_N$ (also assuming $\Delta f(x|\emptyset)>0,\ \forall x\in X$), $\alpha_p$ can be simplified as

$$\alpha_p = 1-\min_{\substack{(B,x):x\in X,\\ B\cup\{x\}\in\mathcal{I}_N}}\left[\frac{\Delta f(x|B)}{\Delta f(x|\emptyset)}\right] = 1+\max_{x\in X}\left[\frac{1}{f(x)}\max_{\substack{B:B\in\mathcal{I}_{N-1}\\ x\notin B}}[-\Delta f(x|B)]\right]. \tag{3.10}$$

Now, evaluating the inner set-optimization problem in (3.10) can be easier depending on the structural properties of the considered application. For example, when $-\Delta f(x|B)$ is submodular with respect to $B\subset X\backslash\{x\}$, an upper bound to the $\alpha_p$ (i.e., a lower bound to $\beta_p$) can be computed by executing a greedy algorithm at every point $x\in X$. Moreover, note that $\beta_p$ will be closer to 1 if $\Delta f(x|B)\simeq\Delta f(x|\emptyset)$ for all $B,x$ where $x\in X, B\cup\{x\}\in\mathcal{I}_N$ (a less restrictive condition than that saw for $\beta_t$).

## 3.3 New Curvature Concepts and Performance Bounds

In this section, as opposed to the existing curvature concepts ($\alpha_t, \alpha_g, \alpha_e$ and $\alpha_p$) reviewed in the previous section, two new curvature concepts are proposed for the class of submodular maximization problems (i.e., (3.1) under Assumption 3.1).

### 3.3.1 Extended Greedy Curvature

The proposed *extended greedy curvature* measure has three variants: $\alpha_{d1}, \alpha_{d2}$ and $\alpha_{d3}$, with respective performance bounds being $\beta_{d1}, \beta_{d2}$ and $\beta_{d3}$. Each of these curvature measures is evaluated based on the information obtained from executing 1, $N$ and $(N+1)$ *additional* greedy iterations (of Alg. 3.1, assuming $n \geq 2N+1$), respectively.

**Extended Greedy Curvature - I:** Upon finishing the $N^{\text{th}}$ greedy iteration where $Y^G$ was found, executing an additional greedy iteration reveals a set of marginal gains:

$$E_1 = \{\Delta f(x_i|Y^G) : x_i \in X \backslash Y^G\}. \tag{3.11}$$

Taking $\alpha_{d1}^j$ as the $j^{\text{th}}$ largest entry in $E_1$, the curvature measure $\alpha_{d1}$ is defined as

$$\alpha_{d1} \triangleq \sum_{j=1}^{N} \alpha_{d1}^j. \tag{3.12}$$

**Lemma 3.1.** *Based on $\alpha_{d1}$ in* (3.12)*, a performance bound $\beta_{d1}$ can be imposed as*

$$\beta_{d1} \triangleq \left[1 + \frac{\alpha_{d1}}{f(Y^G)}\right]^{-1} \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.13}$$

*Proof.* See Appendix A.3.1. □

**Remarks:** The performance bound $\beta_{d1}$ in (3.13) is closer to 1 when $\alpha_{d1} \ll f(Y^G)$. This can be expected to happen when $N$ values are larger and/or submodularity properties are stronger. Moreover, evaluating $\beta_{d1}$ is computationally cheap. Therefore, $\beta_{d1}$ seems to have key strengths of both the elemental curvature and the greedy curvature based performance bounds (i.e., $\beta_e$ and $\beta_g$).

**Numerical Example:** The following Tab. 3.1 shows marginal gain values observed in a example problem where $N = 3$ and $n = 10$ have been used. Here, $Z^i$ represents the set of elements chosen from the ground set at the end of the $i^{\text{th}}$ greedy iteration. Note that four extra greedy iterations have been executed after the required initial 3 greedy iterations. Underlined values in the fourth row ($i = 4$) corresponds to the $\alpha_{d1}^j$ values (thus $\alpha_{d1} = 63.22$) and $f(Y^G) = f(Z^3) = 310.86$. Therefore, the resulting performance bound according to Lemma 3.1 is $\beta_{d1} = 0.831$. For this case, $\beta_f = 0.704$ and $\beta_e = 0.788$ was observed (thus $\beta_{d1}$ is tighter).

**Table 3.1:** Observations from a numerical example.

| $\Delta f(x_j \mid Z^{i-1})$ | | Ground Set $X$ ($x_j \in X$) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ | $f(Z^i)$ |
| Iteration ($i$) | 1 | 140.83 | 160.65 | 138.65 | 161.72 | **186.25** | 168.44 | 148.99 | 168.29 | 155.08 | 178.68 | 186.25 |
| | 2 | 66.40 | 75.56 | 67.58 | 73.25 | - | 79.72 | 68.32 | 76.73 | 73.98 | **82.72** | 268.97 |
| | 3 | 33.43 | 36.10 | 31.18 | 38.78 | - | 39.17 | 38.82 | **41.89** | 39.77 | - | $\underline{\textbf{310.86}}$ |
| | 4 | 19.95 | $\underline{\textbf{21.22}}$ | 18.40 | $\underline{20.78}$ | - | $\underline{21.21}$ | 18.93 | - | 19.62 | - | 332.08 |
| | 5 | 9.60 | - | 9.86 | 11.41 | - | 12.37 | 11.52 | - | **12.52** | - | 344.60 |
| | 6 | 6.58 | - | 5.61 | **7.10** | - | 6.35 | 6.97 | - | - | - | $\underline{\textbf{351.70}}$ |
| | 7 | 3.18 | - | $\underline{3.47}$ | - | - | $\underline{\textbf{3.84}}$ | $\underline{3.34}$ | - | - | - | - |

**Extended Greedy Curvature - II:** Consider a situation where an additional $N$ greedy iterations are executed. Let the obtained final (extended) greedy solution be denoted as $Y^{G2}$ (note that $|Y^{G2}| = 2N$). The proposed new curvature measure $\alpha_{d2}$ is

$$\alpha_{d2} \triangleq \frac{1}{\beta_{f\#}} \left[ f(Y^{G2}) - f(Y^G) \right], \tag{3.14}$$

where $\beta_{f\#}$ is a valid fundamental performance bound for the auxiliary problem

$$Y_{\#}^* = \underset{Y \in \mathcal{I}_{\#}}{\arg\max}\ \Psi(Y), \tag{3.15}$$

where $\Psi(Y) \triangleq f(Y \cup Y^G) - f(Y^G)$ and $\mathcal{I}_{\#} \triangleq \{Y : Y \subseteq X \backslash Y^G, |Y| \leq N\}$. Note that (3.15) is analogous to (3.1). The following lemma gives a candidate value for $\beta_{f\#}$.

**Lemma 3.2.** *The set function* $\Psi(Y)$ *in the problem* (3.15) *is a polymatroid function, and, if* $Y = Y^G_{\#}$ *is a greedy solution to the problem* (3.15)*, then,*

$$\beta_{f\#} = 1 - (1 - \frac{1}{N})^N \leq \frac{\Psi(Y^G_{\#})}{\Psi(Y^*_{\#})}. \tag{3.16}$$

*Proof.* See Appendix A.3.2. □

**Lemma 3.3.** *Based on* $\alpha_{d2}$ *in* (3.14)*, a performance bound* $\beta_{d2}$ *can be imposed as,*

$$\beta_{d2} \triangleq \left[1 + \frac{\alpha_{d2}}{f(Y^G)}\right]^{-1} \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.17}$$

*Proof.* See Appendix A.3.3. □

**Remarks:** Since both (3.13) and (3.17) has a similar format, the corresponding performance bounds will follow $\beta_{d1} \leq \beta_{d2}$ whenever $\alpha_{d1} \geq \alpha_{d2}$.

Note that the components of $\alpha_{d1}$ (i.e., $\{\alpha^i_{d1}\}_{i=1,\ldots,N}$ in (3.12)) are the highest marginal gain values when one additional element is to be added. Therefore, it is possible that these components correspond to a similar, or closely located set of elements in the ground set $X$. Given the form of (A.8) and (A.9), picking such a similar set of elements may result in less tight performance bounds.

In contrast, the components of $\alpha_{d2}$ (see (3.14)) correspond to a set of greedily picked elements. Therefore, it ensures that those elements are more spread-out in the ground set. As a result, it can be expected that $\alpha_{d1} \geq \alpha_{d2}$ resulting $\beta_{d1} \leq \beta_{d2}$ when $N$ is larger or when the submodularity property becomes stronger.

For the given numerical example in Tab. 3.1, $f(Y^{G2}) = 351.7$ (i.e., $f(Z^6)$ as $N = 3$) and $\beta_{f\#} = 0.704$. This results in $\alpha_{d2} = 58.04$. Now, Lemma 3.3 gives $\beta_{d2} = 0.843$ which is tighter than $\beta_{d1}(= 0.831$, given by Lemma 3.1).

**Extended Greedy Curvature - III:** Notice that the curvature measure $\alpha_{d2}$ given in (3.14) can be further decreased if its $\beta_{f\#}$ term can be replaced by a term (say $\beta_{d\#}$) with a higher value (i.e., $\beta_{f\#} \leq \beta_{d\#}$). Since $\beta_{f\#}$ is the fundamental performance bound of the auxiliary problem in (3.15), it is natural to think of using the bound given in Lemma 3.1 to get an improved bound for (3.15). This is the motivation behind the proposing a new curvature measure $\alpha_{d3}$, defined as

$$\alpha_{d3} \triangleq \frac{1}{\beta_{d\#}}\left[f(Y^{G2}) - f(Y^G)\right]. \tag{3.18}$$

where $\beta_{d\#}$ is the performance bound given by the Lemma 3.1 for (3.15). Specifically,

$$\beta_{d\#} = \left[1 + \frac{\alpha_{d1\#}}{\Psi(Y^G_\#)}\right]^{-1} \leq \frac{\Psi(Y^G_\#)}{\Psi(Y^*_\#)},$$

where $\Psi(Y^G_\#) = f(Y^{G2}) - f(Y^G)$ and $\alpha_{d1\#}$ is computed using the marginal gain information from the $(N+1)^{\text{th}}$ additional greedy iteration for (3.15). Hence, in total, $(2N+1)^{\text{th}}$ greedy iterations for (3.1) should be executed to get to the $\alpha_{d1\#}$ value.

**Lemma 3.4.** *Based on $\alpha_{d3}$ in* (3.18)*, a performance bound $\beta_{d3}$ can be imposed as,*

$$\beta_{d3} \triangleq \left[1 + \frac{\alpha_{d3}}{f(Y^G)}\right]^{-1} \leq \frac{f(Y^G)}{f(Y^*)} \tag{3.19}$$

*Proof.* The proof follows the same steps as that of Lemma 3.3 and is, therefore, omitted. □

**Remarks:** The term $\beta_{f\#}$ in (3.14) could have been replaced with any other performance bound that was discussed in Section 3.2. However, since it was shown that the bound given in Lemma 3.3 (and also in Lemma 3.1) works well when $N$ is large or submodularity property is strong, the best choice to replace $\beta_{f\#}$ with is the bound given in Lemma 3.1 (i.e., $\beta_{d\#}$ defined above). Hence, it can be expected that $\beta_{d1} \leq \beta_{d2} \leq \beta_{d3}$ when $N$ increases or when submodularity property becomes stronger.

For the given numerical example in Tab. 3.1, $\Psi(Y^G_{\#}) = 40.84$. Here, $\alpha_{d1\#}$ is computed using the iteration $i = 7$ as $\alpha_{d1\#} = 10.66$ (sum of the underlined values in the 7th row). This respectively results in $\beta_{d\#} = 0.793$, $\alpha_{d3} = 51.50$ and $\beta_{d3} = 0.858$. Note that the bound $\beta_{d3}$ is tighter than both bounds $\beta_{d1}$ and $\beta_{d2}$ computed before.

**Summary:** The following theorem combines the three performance bounds obtained based on the three proposed extended greedy curvature measures discussed so far.

**Theorem 3.3.** *For the submodular maximization problem in* (3.1)*, under assumption 3.1, the greedy solution $Y^G$ given by the Alg. 3.1 has a performance bound $\beta_d$*

$$\beta_d \triangleq \max\{\beta_{d1}, \beta_{d2}, \beta_{d3}\} \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.20}$$

*where, $\beta_{d1}, \beta_{d2}$ and $\beta_{d3}$ are respectively given by Lemmas 3.1, 3.3 and 3.4.*

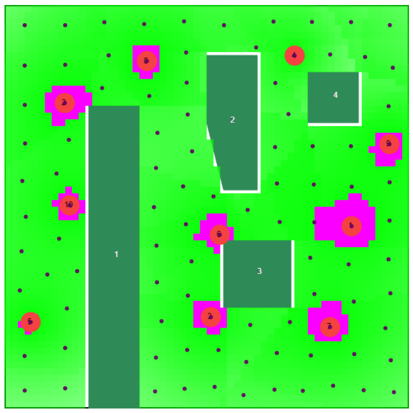
*Proof.* This result directly follows from Lemmas 3.1, 3.3 and 3.4. □

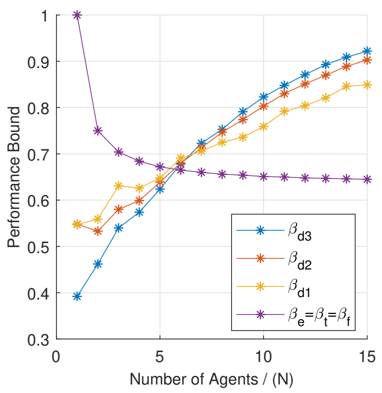
**Remarks:** The proposed extended greedy curvature measure has the same advantages as the elemental curvature measure (reviewed in Section 3.2.3). That is, both methods provide better performance bounds when $N$ is high or submodularity properties are strong. However, recall that the elemental curvature measure fails when some region of the set-objective function is modular. In contrast, the proposed extended greedy curvature concept does not suffer from such a limitation.

Further, in terms of the computational power and the evaluation technique, the proposed extended greedy curvature measure has the same advantages as the greedy curvature measure (reviewed in Section 3.2.2). That is, being computationally cheap and having the ability to evaluate on-line.

Furthermore, having three different versions of the extended greedy curvature based performance bounds such that $\beta_{d1} \leq \beta_{d2} \leq \beta_{d3}$ (valid when $N$ is high or submodularity properties are strong) which respectively requires 1, $N$ and $(N+1)$ additional greedy iterations is also an advantage. This is because, if $\beta_{d1}$ was found to be much lower than the fundamental bound $\beta_f$ in a specific application, one could avoid executing unnecessary additional greedy iterations (that aims to get $\beta_{d2}$, $\beta_{d3}$).

For example, Fig. 3·1 (generated for a multi-agent coverage application) shows that when $N \leq 6$, evaluating $\beta_{d2}$ and $\beta_{d3}$ is of no use as $\beta_{d1}$ is already closer or below the fundamental bound $\beta_f$. However, as $N$ increases beyond $N = 6$, $\beta_{d1}$ becomes much better than $\beta_f$. In such cases, evaluating $\beta_{d2}$ and $\beta_{d3}$ by running a few more additional greedy iterations is profitable as $\beta_{d1} \leq \beta_{d2} \leq \beta_{d3}$.


**(a)** $N = 10$



**(b)**

**Figure 3·1:** Different extended greedy curvature based performance bounds and the fundamental performance bound with respect to the number of agents used (i.e., $N$), for a coverage application scenario.

### 3.3.2 Modularity Based Performance Bound

The proposed extended greedy curvature concept fails when the objective function's modular nature dominates (i.e., when $N$ is low or submodularity properties are weak). However, for such paradigms, the curvature concepts: total curvature, greedy curvature and partial curvature (reviewed in Section 3.2.3) can be used. Nevertheless, out of these three, only the greedy curvature method is computationally cheap (and also on-line). The proposing *modularity based performance bound* aims to outperform the greedy curvature based performance bounds.

Note that when the first greedy iteration is executed, it reveals a set $E_0$

$$E_0 = \{\Delta f(x_i|\emptyset) : x_i \in X\}. \tag{3.21}$$

Taking $\alpha_m^j$ as the $j^{\text{th}}$ largest entry in $E_0$, a new curvature measure $\alpha_m$ is defined as

$$\alpha_m \triangleq \sum_{j=1}^{N} \alpha_m^j. \tag{3.22}$$

Note $\alpha_m^1$ corresponds to the first element of the greedy solution $Y^G$.

**Theorem 3.4.** *Based on the curvature measure $\alpha_m$ in* (3.22)*, a performance bound $\beta_m$ can be imposed as*

$$\beta_m \triangleq \frac{f(Y^G)}{\alpha_m} \leq \frac{f(Y^G)}{f(Y^*)}. \tag{3.23}$$

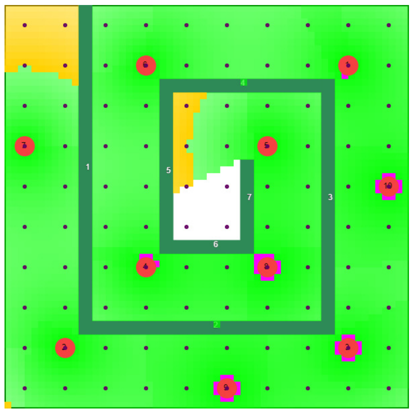
*Proof.* See Appendix A.3.4. □

**Remarks:** The intuition behind the above theorem is to create an upper bound to the set-objective function $f$ based on the simplest submodularity property result:

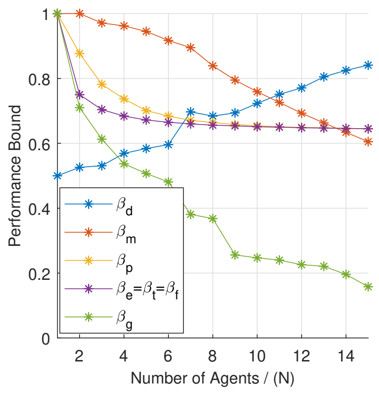
$$f(A) \leq \sum_{i=1}^{k} f(\{a_i\}), \text{ for any } A = \{a_1, a_2, \ldots, a_k\} \subseteq X \tag{3.24}$$

Note that in (3.24), the equality holds when $f$ is modular. Hence, this performance bound (i.e., $\beta_m$ given in (3.23)) is called the "modularity" based performance bound. Note that, in a such modular setting, the curvature measure $\alpha_m$ (which is an upper-bound for $f(Y^*)$) becomes low, resulting a high (tight) performance bound.

This is evident from Fig. 3·2 - an example taken from the class of coverage problems. Note that when $N$ is low (i.e., when $f$ is closer to being modular), the performance bound $\beta_m$ is tighter than $\beta_f$ or any other bounds. This example also shows the complementary nature of the proposed two new performance bounds: $\beta_d$ and $\beta_m$.



(a) $N = 10$



(b)

**Figure 3·2:** Performance Bounds given by different curvature concepts with respect to number of agents $N$, for a coverage application.

## 3.4 Application to Multi-Agent Coverage Control Problems

This section considers the multi-agent coverage problem introduced in Section 2.3 and models it as a submodular maximization problem of the form (3.1). Upon establishing a few key properties of this coverage problem, a distributed greedy algorithm is proposed to find a greedy solution. The existing and new performance bounds discussed in previous sections are applied to characterize such a greedy solution. Finally, theoretical results that enable the application of such performance bounds and numerical results that illustrates the importance of such performance bounds are provided.

### 3.4.1 Set-Function Approach for Multi-Agent Coverage Problems

The multi-agent coverage problem (Zhong and Cassandras, 2011) introduced in Section 2.3 aims to determine the optimal arrangement of $N$ sensor nodes (*agents*) in a given mission space $\Omega \subseteq \mathbb{R}^2$ so as to maximize the probability of detecting randomly occurring events in the mission space. Recall that each agent location is denoted by a continuous variable $s_i \in \mathbb{R}^2$ with $i \in \{1, 2, \ldots, N\}$ and the global state variable is denoted by $\mathbf{s} = [s_1, s_2, \ldots, s_N]^T$. Taking the said coverage objective as a function $H : \mathbb{R}^{N\times 2} \to \mathbb{R}$, the coverage problem can be stated as (identical to (2.39))

$$\begin{aligned} \mathbf{s}^* = \underset{\mathbf{s}}{\text{argmax}}\ H(\mathbf{s}) \\ \text{subject to: } s_i \in F,\ i = 1, 2, \ldots, N, \end{aligned} \tag{3.25}$$

where $F$ is the feasible space such that $F \subseteq \Omega \subset \mathbb{R}^2$ (See also Section 2.3).

The following steps are now used to model this problem as a set function maximization problem of the form (3.1). First, the *ground set* $X = \{x_1, x_2, \ldots, x_n\}$ is created by discretizing the feasible space $F$ (uniformly or randomly, such that each $x_i \in F$). Next, the *set-variable* is defined as $S = \{s_1, s_2, \cdots, s_N\}$ with each $s_i \in X$. Finally, since the number of agents to be deployed are limited to $N$, a *uniform matroid constraint* of rank $N$: $S \in \mathcal{I}_N$ where $\mathcal{I}_N = \{A : A \subseteq X, |A| \leq N\}$ is introduced. In all, the corresponding set function maximization problem is (similar to (3.1))

$$S^* = \underset{S\in\mathcal{I}}{\text{argmax}}\ H(S), \tag{3.26}$$

and the underlying pair $\mathcal{M} = (X, \mathcal{I})$ is a uniform matroid of rank $N$ (i.e., $\mathcal{I} = \mathcal{I}_N$).

### 3.4.2 Properties of Set-Coverage Objective $H(S)$

The exact form of the *set-coverage function* $H(S)$ in (3.26) comes directly from $H(\mathbf{s})$ in (3.25) (i.e., from (2.38)) and can be written as

$$H(S) = \int_F R(x)(1 - \prod_{\forall s_i \in S}(1 - p_i(x, s_i)))dx. \tag{3.27}$$

Recall that $p_i(x, s_i)$ represents the detection probability of an event occurring at a location $x \in F$ by the agent $i$ stationed at $s_i \in F$.

**Theorem 3.5.** *(Sun et al., 2019) The set-coverage function $H(S)$ in* (3.27) *is a polymatroid function (i.e. monotone, submodular, and normalized).*

**Corollary 3.1.** *The set-coverage function $H(S)$ in* (3.27) *is such that:*
*(i)* $H(A \cup B) \leq H(A) + H(B), \quad \forall A, B \subseteq X$*,*
*(ii)* $H(A \cup B \cup C) + H(A) \leq H(A \cup B) + H(A \cup C), \quad \forall A, B, C \subseteq X$*.*

*Proof.* See Appendix A.3.5 □

Notice that the first result in Corollary 3.1 takes the form of the famous triangle inequality. This result can be used to establish a simple upper-bound for $H(S)$ as $H(S) \leq \sum_{i=1}^{N} H(\{s_i\})$ (useful when imposing performance bounds and normalizing).

### 3.4.3 A Brief Summary of (Sun et al., 2019)

The work in (Sun et al., 2019) has proposed the greedy algorithm in Alg. 3.2 to construct a greedy solution (denoted by $S^G$) for the coverage problem in (3.26). Moreover, (Sun et al., 2019) has exploited Theorem 3.5 to establish three performance bounds: $\beta_f$ (Theorem 3.1), $\beta_t$ (3.3) and $\beta_e$ (3.7) for the obtained greedy solution $S^G$.

**Algorithm 3.2** The greedy algorithm proposed in (Sun et al., 2019) to solve (3.26).

```
1: $Z := \emptyset$; $i = 0$;
2: while $i \leq N$ do
3:     $z_i = \underset{z:(S \cup \{z\}) \in \mathcal{I}}{\operatorname{argmax}} H(S \cup \{z\})$;
4:     $Z = Z \cup \{z_i\}$;
5: end while
6: Return $S^G = Z$;
```

In contrast to (Sun et al., 2019), note that, in this chapter, two new performance bounds are adopted from the literature: $\beta_g$ (3.5) and $\beta_p$ (3.9), while also proposing two completely new performance bounds: $\beta_d$ (3.20) and $\beta_m$ (3.23). Furthermore, it will be proven that Alg. 3.2 can be executed in a distributed manner. The following two subsections provide several theoretical results required to make these contributions.

### 3.4.4 Properties of the Marginal Coverage Gain Function $\Delta H(s_i|A)$

From Def. 3.1, the marginal gain function of the set-coverage function (3.27) is $\Delta H(s_i|A) \triangleq H(A\cup\{s_i\})-H(A)$ and it is called the *marginal coverage gain* function.

**Definition 3.11.** *In the considered multi-agent coverage problem setting, (recall that,) two agents $i$ and $j$ are considered as "neighbors" if there exists some $x \in F$ such that $p_i(x,s_i)p_j(x,s_j) > 0$. Moreover, $B_i$ and $\bar{B}_i$ respectively represents the set of neighbors and the closed neighborhood. Also, $\bar{s}_i = \{s_j : j \in \bar{B}_i\}$ is the neighborhood state.*

**Theorem 3.6.** *The marginal coverage gain function $\Delta H(s_i|A) = H(A\cup\{s_i\})-H(A)$ can be evaluated using only the local information (i.e., $\bar{s}_i$) at an agent $i$ as*

$$\Delta H(s_i|A) = H_i(\bar{s}_i) \triangleq \int\limits_F R(x)p_i(x,s_i)\prod_{j\in B_i}(1-p_j(x,s_j)))dx, \tag{3.28}$$

*where $B_i$ is the set of neighbors in the agent set $A$.*

*Proof.* See Appendix A.3.6. □

**Corollary 3.2.** *Algorithm 3.2 can be executed equivalently in a distributed manner by replacing its step 3:*

$$\Big\{z_i = \underset{z:(S\cup\{z\})\in\mathcal{I}}{\operatorname{argmax}}\ H(S\cup\{z\})\Big\} \quad \textit{with} \quad \Big\{z_i = \underset{z\in X\backslash S}{\operatorname{argmax}}\ \Delta H(z|S)\Big\}. \tag{3.29}$$

*Proof.* The equivalence is clear from observing: (i) $H(S\cup\{z\}) = \Delta H(z|S) + H(S)$, (ii) $H(S)$ is independent of $z$ and (iii) $\{z : (S\cup\{z\}) \in \mathcal{I}\} \equiv \{z \in X\backslash S\}$. The fact that $\Delta H(z|S)$ can be evaluated by an agent (at $z$) only using its neighborhood state (from Theorem 3.6) implies the distributed nature of the resulting greedy algorithm. □

**Theorem 3.7.** *The marginal coverage gain function $\Delta H(s_i|A)$ is a non-increasing supermodular set function in $A \subseteq (X\backslash\{s_i\})$ for some fixed $s_i \in X$.*

*Proof.* See Appendix A.3.7. □

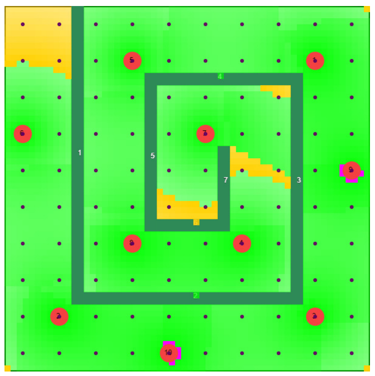
The monotonicity property established above implies that whenever a new agent is added, all the local objective functions (i.e., marginal coverage gain functions) of the existing agents will decrease (or remain the same). Apart from establishing such an underlying structural property of the coverage problem, this theorem also enables efficient evaluation of the partial curvature metric (See (3.10) and the corresponding discussion). Several similar theoretical results that provide intuitions about the coverage problem and also enables the application of the partial curvature concept (i.e., Theorem 3.2) are discussed in appendix C.1.

### 3.4.5 Numerical Results

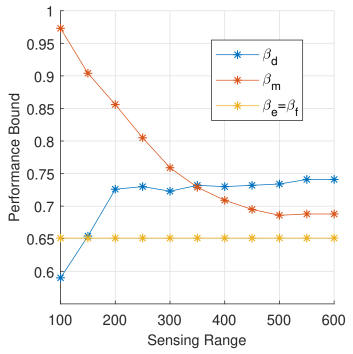
The proposed greedy algorithm (Alg. 3.2 with (3.29)), the two newly adopted performance bounds: $\beta_g$ (3.5) and $\beta_p$ (3.9), the two newly proposed performance bounds: $\beta_d$ (3.20) and $\beta_m$ (3.23) and the existing other performance bounds: $\beta_e$ (3.7), $\beta_t$ (3.3) and $\beta_f$ (given in Theorem 3.1), were all implemented for the considered class of multi-agent coverage problems in an interactive JavaScript-based simulator which is available at `http://www.bu.edu/codes/simulations/shiran27/CoverageFinal/` (same as in Chapter 2). It may be used by the reader to reproduce the reported results and also to try different new problem configurations.

In particular, the main focus here is to study the behavior of different performance bounds under different coverage problem configurations. In each experiment, one of the three parameters: (i) sensing range $\delta$, (ii) sensing decay rate $\lambda$ or (iii) the number of deployed agents $N$, was varied while keeping the other two fixed. More detailed definitions of sensing parameters $\delta$ and $\lambda$ can be found in Section 2.3. However, for now, it is sufficient to know that as $\delta$ increases (or $\lambda$ decreases), the sensing capability/power of an agent also increases. For convenience, each graph has been drawn so that along its $x$-axis, the sensing capability of the agents' increases. Also, note that whenever only a few of the said performance bounds have been drawn in a graph, it means the other performance bounds were redundant (no better than $\beta_f$).

Across all the simulation results shown, the proposed extended greedy curvature based performance bound $\beta_d$ has shown the best results when the agents have high sensing capabilities. The proposed modularity based performance bound $\beta_m$ has been effective when the agents have low sensing capabilities and when the mission space has more obstacles (like in Fig. 3·2, 3·3, 3·4, 3·5 and 3·6). In cases where the agents have low sensing capabilities and fewer obstacles in the mission space (like in Fig. 3·7 and 3·8), the partial curvature based performance bound $\beta_p$ has delivered the best performance bounds.

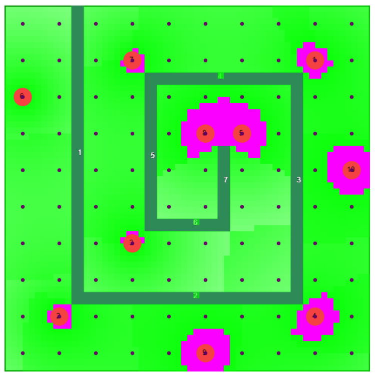


**(a)** $\delta = 250$

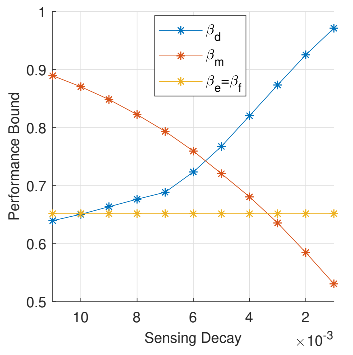


**(b)**

**Figure 3·3:** Performance Bound Vs Sensing Range; Maze environment with $N = 10$ and Sensing Decay $\lambda = 0.006$.

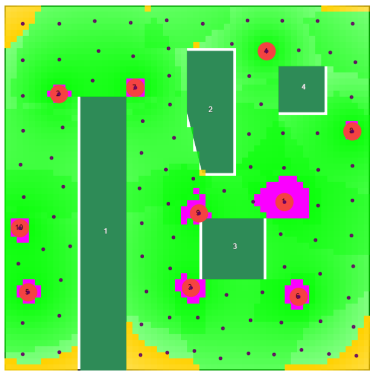


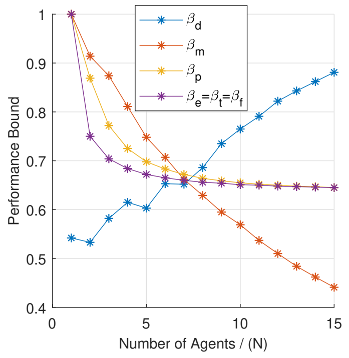
**(a)** $\lambda = 0.004$



**(b)**

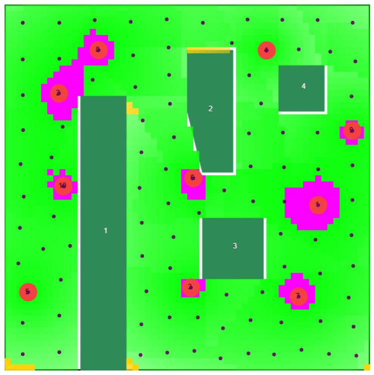
**Figure 3·4:** Performance Bound Vs Sensing Decay; Maze environment with $N = 10$ and Sensing Range $\delta = 300$.

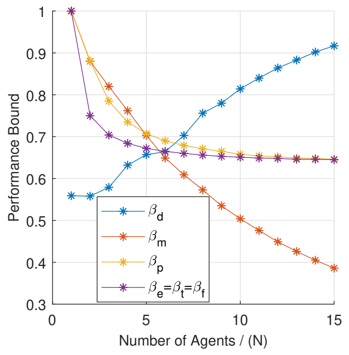

**(a)** $N = 10$



**(b)**

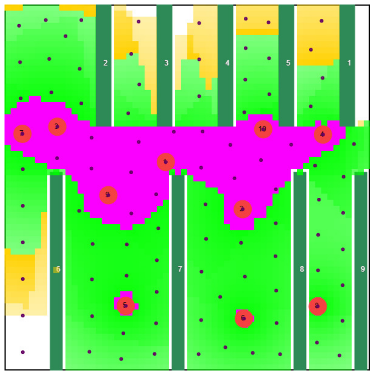
**Figure 3·5:** Performance Bound Vs Number of Agents; General environment with Sensing decay $\lambda = 0.006$ and Sensing Range $\delta = 200$.

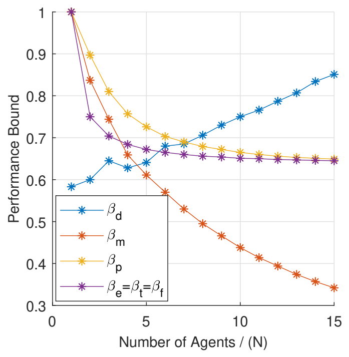

**(a)** $N = 10$



**(b)**

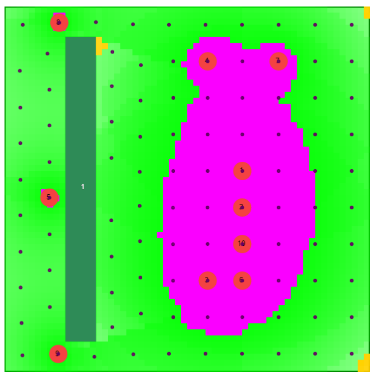
**Figure 3·6:** Performance Bound Vs Number of Agents; General environment with Sensing decay $\lambda = 0.006$ and Sensing Range $\delta = 300$.

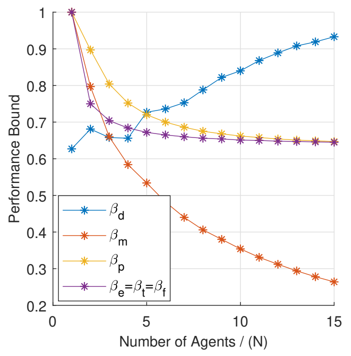

**(a)** $N = 10$



**(b)**

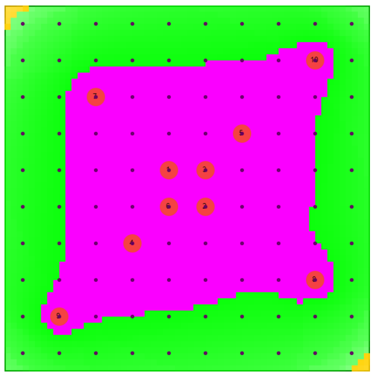
**Figure 3·7:** Performance Bound Vs Number of Agents; Room environment with Sensing decay $\lambda = 0.006$ and Sensing Range $\delta = 300$.

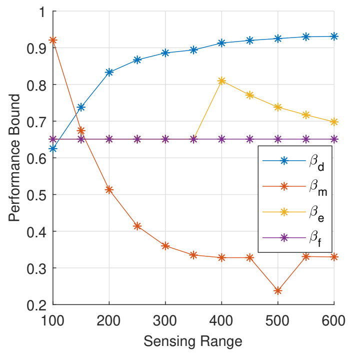

(a) $N = 10$



(b)

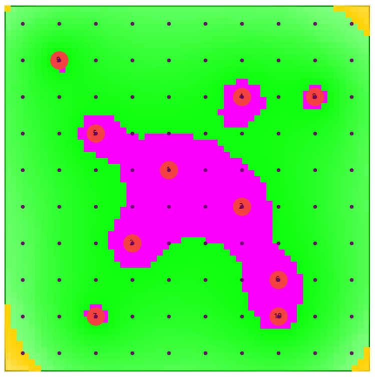
**Figure 3·8:** Performance Bound Vs Number of Agents; Narrow environment with Sensing decay $\lambda = 0.006$ and Sensing Range $\delta = 300$.

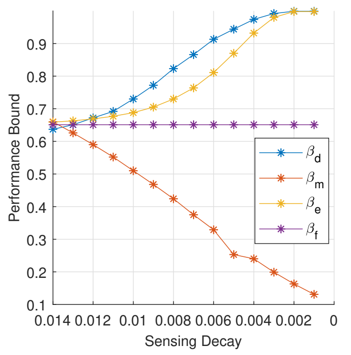

(a) $\delta = 400$



(b)

**Figure 3·9:** Performance Bound Vs Sensing Range; Blank environment with $N = 10$ and Sensing Decay $\lambda = 0.006$.


(a) $\delta = 400$



(b)

**Figure 3·10:** Performance Bound Vs Sensing Decay; Blank environment with $N = 10$ and Sensing Range $\delta = 400$.

## 3.5 Summary

The use of a greedy initialization technique is a practical approach to overcome the issue of converging to poor local optima in cooperative multi-agent optimization problems. In particular, when the objective function is submodular, such a greedy initialization technique can also provide performance bound guarantees for the obtained solutions (initial greedy or any other subsequent solution (Sun et al., 2020)). Such a performance bound is highly valued as it indicates the closeness to the global optimal. For the class of submodular maximization problems, several existing performance bounds were reviewed. For the same class of problems, computationally efficient two new performance bounds were also proposed. A class of coverage problems was modeled as a class of submodular maximization problems so as to study the effectiveness of different performance bounds. Obtained numerical results show that the proposed new performance bounds provide significant improvements compared to the state-of-the-art performance bounds established for the coverage problem.

# Chapter 4

# Greedy Initialization for Persistent Monitoring in Networks

This chapter of the thesis addresses the issue of local optima arising in *persistent monitoring on networks* problems as introduced in Section 1.3.2. For PMN problems introduced in Section 1.3.1, a class of distributed threshold-based (parametric) controllers has been proposed in (Zhou et al., 2019) along with an on-line gradient technique to determine the optimal threshold values. However, due to the problem's non-convexity, this approach often leads to a poor local optima highly dependent on the initial thresholds used. To overcome this initialization challenge, a computationally efficient off-line greedy technique is developed based on the asymptotic analysis of the network system. Extensive numerical results show that such initial thresholds are almost immediately (locally) optimal or quickly lead to optimal values.

This chapter is organized as follows. Section 4.1 provides the problem formulation and reviews the *threshold-based control* approach proposed in (Zhou et al., 2019). Section 4.2 includes the asymptotic analysis and a candidate threshold initialization technique, assuming the underlying PMN problem is single-agent and the network is sufficiently dense. Next, Section 4.3 generalizes the asymptotic analysis and the threshold initialization technique proposed in Section 4.2 to any network (still assuming a single-agent PMN scenario). Subsequently, Section 4.4 generalizes the proposed threshold initialization technique to multi-agent systems. Finally, Section 4.5 presents several numerical examples and performance comparisons with respect to the solution in (Zhou et al., 2019) while Section 4.6 concludes the chapter.

## 4.1 Problem Formulation

Consider an $n$-dimensional mission space with $M$ targets in the set $\mathcal{T} = \{1, 2, \ldots, M\}$ and $N$ agents in the set $\mathcal{A} = \{1, 2, \ldots, N\}$ where $M \geq N$. Each target $i \in \mathcal{T}$ is located at a fixed position $Y_i \in \mathbb{R}^n$ and each agent $a \in \mathcal{A}$ is allowed to move in the mission space where its trajectory is denoted by $\{s_a(t) \in \mathbb{R}^n, t \geq 0\}$. As proposed in (Zhou et al., 2019) and as shown in Fig. 4·1, a network topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is embedded to this mission space such that the graph vertices represent the targets (i.e., $\mathcal{V} = \mathcal{T}$) and the graph edges represent the inter-target trajectory segments available for agents to travel (i.e., $\mathcal{E} =\subseteq \{(i,j) : i, j \in \mathcal{V}\}$). The shape of each trajectory segment $(i,j) \in \mathcal{E}$ can be considered as a result of a lower level optimal control problem that minimizes the travel-time that an agent takes to go from target $i$ to target $j$ while accounting for potential constraints in the mission space and agent dynamics. In this thesis, each trajectory segment $(i,j) \in \mathcal{E}$ is assumed to have a fixed such optimal travel-time value $\rho_{ij} \in \mathbb{R}_{\geq 0}$. Based on $\mathcal{E}$, the *neighbor set* $\mathcal{N}_i$ of target $i \in \mathcal{V}$ is defined as $\mathcal{N}_i \triangleq \{j : (i,j) \in \mathcal{E}\}$. Note also that the target locations $\{Y_i : i \in \mathcal{V}\}$, initial agent locations $\{s_a(0) : a \in \mathcal{A}\}$ and travel-time values $\{\rho_{ij} : (i,j) \in \mathcal{E}\}$ are prespecified.

**Target Model:** Each target $i \in \mathcal{V}$ has an associated *uncertainty state* $R_i(t) \in \mathbb{R}$ which follows the dynamics (Zhou et al., 2019):

$$\dot{R}_i(t) = \begin{cases} 0 & \text{if } R_i(t) = 0 \text{ and } A_i \leq B_i N_i(t), \\ A_i - B_i N_i(t) & \text{otherwise,} \end{cases} \tag{4.1}$$

where $R_i(0)$ is prespecified, $A_i \in \mathbb{R}_{\geq 0}$ is the uncertainty growth rate, $B_i \in \mathbb{R}_{>0}$ is the uncertainty removal rate by an agent and $N_i(t) = \sum_{a=1}^{N} \mathbf{1}\{s_a(t) = Y_i\}$ is the number of agents present at target $i$ at time $t$. Simply, (i) $R_i(t)$ increases at a rate $A_i$ when no agent is visiting it, (ii) $R_i(t)$ decreases at a rate $B_iN_i(t) - A_i$ when $N_i(t) > 0$, and (iii) $R_i(t) \geq 0,\ \forall t \geq 0$. As pointed out in (Zhou et al., 2019), (4.1) has an attractive queueing system interpretation where $A_i$, and $B_iN_i(t)$ can be thought of as an arrival rate and a controllable service rate respectively of a node ($i$) in a queueing network .

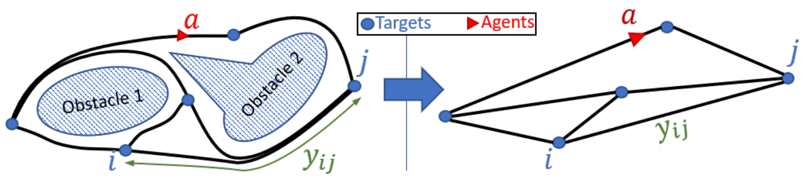


**Figure 4·1:** The network abstraction.

**Agent Model:** In some persistent monitoring models (Zhou et al., 2018), each agent $a \in \mathcal{A}$ is assumed to have a finite sensing range $r_a > 0$ that allows it to decrease $R_i(t)$, $i \in \mathcal{V}$ whenever $\|s_a(t) - Y_i\| \leq r_a$. However, the approach used in (Zhou et al., 2019) is followed here where $r_a = 0$ is assumed and $N_i(t)$ is used to replace the role of the joint detection probability of a target $i$ by the agents.

**Objective Function:** The objective of this persistent monitoring system is to control the team of agents to minimize a measure of *mean system uncertainty* $J_T$ where

$$J_T \triangleq \frac{1}{T}\int_0^T \sum_{i=1}^{M} R_i(t)dt. \tag{4.2}$$

Based on the target dynamics (4.1), to minimize the objective $J_T$ (4.2), it is intuitive that each agent has to *dwell* (i.e., remain stationary) only at targets that it visits in its trajectory. Moreover, based on the embedded network topology $\mathcal{G}$ that constrains the agent motion, it is clear that when an agent $a \in \mathcal{A}$ leaves a target $i \in \mathcal{V}$ its next target would be some $j \in \mathcal{N}_i$ that is only reachable by *traveling* on the edge $(i,j) \in \mathcal{E}$ for a time duration of $\rho_{ij}$. Each time an agent $a \in \mathcal{A}$ arrives at a target $i \in \mathcal{V}$, it has to determine a *dwell-time* $\tau_i^a \in \mathbb{R}_{\geq 0}$ and a *next-visit* target $v_i^a \in \mathcal{N}_i$. Therefore, for the set of agents, the optimal control solution that minimizes the objective $J_T$ takes the form of a set of optimal dwelling times and next-visit target sequences. Determining such an optimal solution is a challenging task even for the simplest PMN problem configurations due to the nature of the involved search space.

**Threshold-Based Control Policy:** To address this challenge, the TCP proposed in (Zhou et al., 2019) is adopted in this chapter. Under this TCP, each agent $a \in \mathcal{A}$ makes its decisions by adhering to a set of pre-specified parameters denoted by $\Theta^a \in \mathbb{R}^{M\times M}$ which serve as thresholds on target uncertainties. The $(i,j)^{\text{th}}$ parameter in $\Theta^a$ matrix is denoted as $\theta_{ij}^a \in \mathbb{R}_{\geq 0}$. The set of neighbors of a target $i$ that violate their thresholds (called *active neighbors*) when agent $a$ is in $i$ at time $t$ is defined as

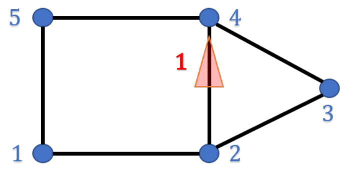
$$\mathcal{N}_i^a(t) \triangleq \{j : R_j(t) > \theta_{ij}^a,\ j \in \mathcal{N}_i\} \subseteq \mathcal{N}_i. \tag{4.3}$$

Assume an agent $a$ arrives at target $i$ at a time $t = t'$. Then, the dwell-time $\tau_i^a$ to be spent at target $i$ is determined by: (i) the diagonal element $\theta_{ii}^a$ based on the *threshold satisfaction* condition $R_i(t) < \theta_{ii}^a$ and (ii) the *active neighbor existence* condition $|\mathcal{N}_i^a(t)| > 0$ at $t = t' + \tau_i^a$ (recall that $|\cdot|$ is the cardinality operator). Subsequently, agent $a$'s next-visit target $v_i^a$ is chosen from the set of active neighbors $\mathcal{N}_i^a(t) \subseteq \mathcal{N}_i$ using the off-diagonal thresholds $\{\theta_{iv}^a : v \in \mathcal{N}_i^a(t)\}$ at $t = t' + \tau_i^a$. Formally,

$$\begin{aligned} \tau_i^a &\triangleq \underset{\tau \geq 0}{\operatorname{arginf}}\ \mathbf{1}\left\{[R_i(t'+\tau) < \theta_{ii}^a]\ \&\ [|\mathcal{N}_i^a(t'+\tau)| > 0]\right\}, \\ v_i^a &\triangleq \underset{v \in \mathcal{N}_i^a(t'+\tau_i^a)}{\operatorname{argmax}}\ \left\{R_v(t'+\tau_i^a) - \theta_{iv}^a\right\}. \end{aligned} \tag{4.4}$$

While the first condition in the $\tau_i^a$ expression in (4.4) ensures that agent $a$ will dwell at target $i$ until at least its own uncertainty $R_i(t)$ drops below $\theta_{ii}^a$, the second condition ensures that when agent $a$ is ready to leave target $i$ there will be at least one neighbor $v \in \mathcal{N}_i$ whose uncertainty $R_v(t)$ has increased beyond the threshold $\theta^a_{iv}$. The $v^a_i$ expression in (4.4) implies that $v^a_i$ is the neighboring target of $i$ chosen from the set $\mathcal{N}^a_i(t' + \tau^a_i) \subseteq \mathcal{N}_i$ with the largest threshold violation. In all, the update equations in (4.4) define each agent's dwell-time and next-visit decision sequence under the TCP.

A key advantage of this TCP approach is that based on (4.3) and (4.4), each agent now only needs to use the neighboring target state information. Thus, each agent operates in a *distributed* manner. An example target topology and an agent threshold matrix are shown in Fig. 4·2. Note that when certain edges are missing in the graph, the respective off-diagonal entries in $\Theta^a$ are irrelevant and hence denoted by $\theta^a_{ij} = \infty$.



$$\Theta^1 = \begin{bmatrix} \theta^1_{11} & \theta^1_{12} & \infty & \infty & \theta^1_{15} \\ \theta^1_{21} & \theta^1_{22} & \theta^1_{23} & \theta^1_{24} & \infty \\ \infty & \theta^1_{32} & \theta^1_{33} & \theta^1_{34} & \infty \\ \infty & \theta^1_{42} & \theta^1_{43} & \theta^1_{44} & \theta^1_{45} \\ \theta^1_{51} & \infty & \infty & \theta^1_{54} & \theta^1_{55} \end{bmatrix}$$

**Figure 4·2:** An example target topology with five targets and one agent with its threshold parameters.

**Discrete Event System View:** Under the described TCP, the behavior of the PMN system is fully defined by $\mathcal{U}(\Theta) = \{(\tau^a_{i(l)}(\Theta^a), v^a_{i(l)}(\Theta^a)) : l \in \mathbb{Z}_{>0},\ a \in \mathcal{A}\}$, i.e., the set of agent decision sequences, where $\Theta \in \mathbb{R}^{M\times M\times N}$ is the collection of all agent threshold matrices and $i(l)$ is the $l^{\text{th}}$ target visited by agent $a$. Following from (4.4), the PMN system is a discrete event system (DES) (Cassandras and Lafortune, 2010) where the *event set* consists of: (i) agent arrivals and departures at/from targets, (ii) instances where a target uncertainty reaches 0 from above, and (iii) the 'start' and the 'end' events triggered respectively at times $t = 0$ and $t = T$. The sequence of *event times* observed is denoted as $\{t^k : k \in \{0, 1, \ldots, K\}\}$ with $t^0 = 0$ and $t^K = T$.

Since the behavior of the PMN system is dependent on the used TCP $\Theta$, the objective $J_T$ in (4.2) is also dependent on $\Theta$. Therefore, within this TCP class of agent controllers, the aim is to determine an optimal TCP (OTCP) $\Theta^*$ such that

$$\Theta^* = \underset{\Theta \geq \mathbf{0}}{\operatorname{argmin}}\ J_T(\Theta) = \frac{1}{T}\sum_{i=1}^{M}\sum_{k=0}^{K}\int\limits_{t^k}^{t^{k+1}} R_i(t)dt. \tag{4.5}$$

Differentiating $J_T(\Theta)$ w.r.t. $\Theta$ gives $\nabla J_T(\Theta) = \frac{1}{T}\sum_{i=1}^{M}\sum_{k=0}^{K}\int_{t^k}^{t^{k+1}} \nabla R_i(t)dt$, where $\nabla \equiv \frac{\partial}{\partial \Theta}$. As shown in (Zhou et al., 2019), it is easy to see that $\nabla J_T(\Theta)$ reduces to $\nabla J_T(\Theta) = \frac{1}{T}\sum_{i=1}^{M}\sum_{k=0}^{K} \nabla R_i(t^k)(t^{k+1}-t^k)$. The solution proposed in (Zhou et al., 2019) uses IPA (Cassandras et al., 2010) to evaluate $\nabla R_i(t^k)$ terms (hence $\nabla J_T(\Theta)$) in an on-line distributed manner. This enables the use of a gradient descent algorithm:

$$\Theta^{(l+1)} = \left[\Theta^{(l)} - \beta^{(l)}\nabla J_T(\Theta^{(l)})\right]^+ \tag{4.6}$$

to update the TCP $\Theta$ iteratively ($[\cdot]^+ = \max\{0,\cdot\}$). The step size $\beta^{(l)}$ is selected so that it diminishes with $l$ following the standard conditions (Bertsekas, 2016).

**Initialization $\Theta^{(0)}$:** In (Zhou et al., 2019), a randomly generated set of initial thresholds has been used as $\Theta^{(0)}$ for (4.6). Due to the non-convexity of the objective function (4.5), the resulting $\Theta^{(l)}$ when (4.6) converges is a local minimum that depends heavily on $\Theta^{(0)}$. Hence, a carefully selected high-performing $\Theta^{(0)}$ can be expected to provide significant improvements over the local minimum obtained from randomly selected $\Theta^{(0)}$. Motivated by this idea, first, structural properties of the underlying PMN system are investigated. That knowledge is then used to construct a candidate for $\Theta^{(0)}$.

**Overview of the PMN Solution:** As proven in (Zhou et al., 2019), in a single-agent PMN system, it is optimal to make the target uncertainty $R_i(t) = 0$ on each visit of agent $a$ at target $i$. In other words, in the OTCP, $\theta_{ii}^a = 0$. Moreover, empirical results in (Zhou et al., 2019) provide some intuition about high-performing agent behaviors: (i) after a brief initial transient phase, each agent converges to a (steady-state) periodic behavior where it cycles across a fixed subset of targets, and, (ii) in this steady state, agents do not tend to share targets with other agents.

These observations are exploited here to efficiently construct a high-performing (favorable) set of agent trajectories so that it can be translated into a better candidate TCP for $\Theta^{(0)}$ in (4.6) compared to a randomly generated $\Theta^{(0)}$. It is clear that such a favorable set of agent trajectories takes the form of a non-overlapping set of *target-cycles* on the given graph. This non-overlapping property implies that if a solution for the single-agent PMN problem is developed, it can be extended to multi-agent PMN problems using appropriate graph partitioning and assignment techniques.

Inspired by this discussion, the proposed PMN solution follows the steps outlined in Alg. 4.1. Note that its Step 6 has already been discussed. A key step of Alg. 4.1 is Step 2, as it requires a technique to find a high-performing agent trajectory (a target-cycle) on a given partition of the graph. In fact, in single-agent PMN problems, only Steps 2, 5 and 6 of Alg. 4.1 should be executed. Hence, in the following Sections 4.2 and 4.3, it is assumed that only one agent is available (i.e., $N = 1$) and a PMN solution is developed by discussing the details of Steps 2 and 5 of Alg. 4.1. In particular, Section 4.2 assumes the network to be sufficiently dense and Section 4.3 relaxes that assumption. The subsequent Section 4.4 extends the proposed solution to multi-agent problems (i.e., $N > 1$) by discussing the details of Steps 1, 3 and 4 of Alg. 4.1.

## 4.2 Single-Agent PMN Solution - Part I

This section focuses only on single-agent PMN problems on *sufficiently dense* graphs. More precisely, a graph is said to be "sufficiently dense" if it is *bi-triangular*.

**Algorithm 4.1** The main steps of the PMN solution.

```
   Input: (i) Target topology (graph) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and (ii) Set of agents $\mathcal{A}$.
   Output: A locally optimal TCP candidate for $\Theta^*$ in (4.5).
1: Partition the given graph $\mathcal{G}$ into $N$ sub-graphs $\{\mathcal{G}_a\}_{a\in\mathcal{A}}$.
2: Find a high-performing agent trajectory in each sub-graph.
3: Refine the sub-graphs along with the agent trajectories.
4: Assign agents to the determined refined agent trajectories (on respective sub-
   graphs) based on initial agent locations.
5: Obtain the corresponding TCP as $\Theta^{(0)} = \{\Theta^{a(0)} : a \in \mathcal{A}\}$.
6: Use $\Theta^{(0)}$ in (4.6) and update $\Theta^{(l)}$ using IPA gradients (Zhou et al., 2019).
```

**Definition 4.1.** *A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| > 3$ is **bi-triangular** if for all $(i, j) \in \mathcal{E}$ there exists $k, l \in \mathcal{V}$ such that $(i, k), (k, j) \in \mathcal{E}$, $(i, l), (l, j) \in \mathcal{E}$, and $k \neq l$.*

An example and a counter example for a bi-triangular graph can be seen in Figs. 4·18(a) and 4·15(a), respectively. The conditions assumed in this section are formally stated in the following assumption (which will be relaxed in subsequent sections).

**Assumption 4.1.** *(i) Only one agent is available (i.e., $\mathcal{A} = \{a\}$) and (ii) The given target topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is bi-triangular.*

Due to Assumption 4.1, in this section, first, a high-performing target-cycle is determined in the given graph $\mathcal{G}$ using a greedy scheme. Such a target-cycle is then transformed to a TCP $\Theta^{(0)}$ for the subsequent use in the gradient descent process (4.6). Note that these steps correspond to Steps 2, 5 and 6 of Alg. 4.1, respectively.

### 4.2.1 Analysis of an Unconstrained Target-Cycle

A *target-cycle* is formally defined as a finite sequence of targets selected from $\mathcal{V}$ such that the corresponding sequence of edges also exists in $\mathcal{E}$. An *unconstrained target-cycle* is a target-cycle with no target on it being repeated. The set of all possible unconstrained target-cycles on the graph $\mathcal{G}$ is denoted by $\mathcal{C}$. A generic unconstrained target-cycle in $\mathcal{C}$ is denoted by $\Xi_i = \{i_1, i_2, \ldots, i_m\} \subseteq \mathcal{V}$, where $i_j \in \mathcal{V}$, $\forall j \in \{1, 2, \ldots, m\}$ and $m = |\Xi_i| \leq M$. The corresponding sequence of edges (fully defined by $\Xi_i$) is denoted by $\xi_i = \{(i_m, i_1), (i_1, i_2), \dots, (i_{m-1}, i_m)\} \subseteq \mathcal{E}$.

Since the aim is to greedily construct a target-cycle that results in a low mean system uncertainty value (i.e., $J_T$ in (4.2)), an assessment criterion for any given arbitrary target-cycle (say $\Xi_i$) is required. Therefore, the metric: *steady state mean cycle uncertainty* is defined as $J_{ss}(\Xi_i)$ where:

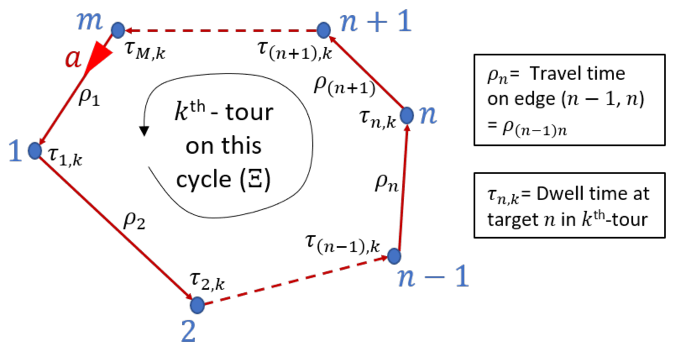
$$J_{ss}(\Xi_i) \triangleq \lim_{T\to\infty} \frac{1}{T} \int_0^T \sum_{j\in\Xi_i} R_j(t)dt. \tag{4.7}$$

A computationally efficient off-line method to evaluate $J_{ss}(\Xi_i)$ for any $\Xi_i \in \mathcal{C}$ is proposed next. First, for notational convenience, $\Xi_i$ and its targets are relabeled as $\Xi = \{1, 2, \dots, n, n+1, \dots, m\}$ by omitting the subscript $i$ (see Fig. 4·3). Then, the following assumption is made regarding the agent's behavior on a target-cycle.

**Assumption 4.2.** *After visiting a target $n \in \Xi$, the agent will leave it if and only if the target uncertainty $R_n$ reaches zero.*

This assumption is partially motivated by the aforementioned theoretical result in (Zhou et al., 2019). Nevertheless, since the main focus here is to initialize (4.6), this potential sub-optimality will be compensated by the eventual use of (4.6).

A *tour* on the target-cycle $\Xi$ (shown in Fig. 4·3) starts/ends when the agent leaves the last target $m$ to reach target 1. The dwell-time spent on a target $n \in \Xi$ when the



$m$
$n+1$
$a$
$\tau_{M,k}$
$\tau_{(n+1),k}$
$\rho_1$
$\rho_{(n+1)}$
$k^{\text{th}}$ - tour on this cycle ($\Xi$)
$\tau_{n,k}$
$n$
1
$\tau_{1,k}$
$\rho_n$
$\rho_2$
$\tau_{(n-1),k}$
$\tau_{2,k}$
$n-1$
2
$\rho_n$= Travel time on edge $(n-1, n)$ = $\rho_{(n-1)n}$
$\tau_{n,k}$= Dwell time at target $n$ in $k^{\text{th}}$-tour

**Figure 4·3:** A generic single-agent unconstrained target-cycle $\Xi$.

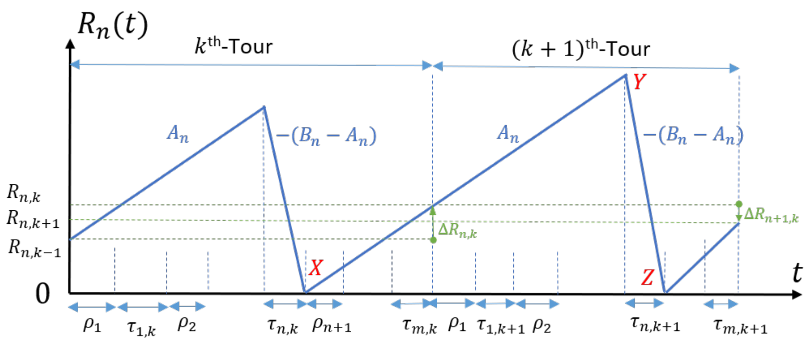


**Figure 4·4:** Variation of target uncertainties during agent tours.

agent is in its $k^{\text{th}}$ tour on $\Xi$ is denoted as $\tau^a_{n,k}$ and the travel-time spent on an edge $(n-1,n) \in \mathcal{E}$ is $\rho_{(n-1)n}$ by definition. Without any ambiguity, the notation $\tau_{n,k}$ and $\rho_n$ (with $\rho_1 = \rho_{m1}$) is used to represent these two quantities respectively. Moreover, target $n$'s uncertainty level at the end of the $k^{\text{th}}$ tour is denoted by $R_{n,k}$. Under this notation, the trajectory of the target uncertainty $R_n(t)$ over $k^{\text{th}}$ and $(k+1)^{\text{th}}$ tours is shown in Fig. 4·4. The geometry of the $XYZ$ triangle shown in Fig. 4·4 can be used to derive the dynamics of target $n$'s dwell-time $\tau_{n,k}$ (w.r.t. $k$) as

$$(B_n - A_n)\tau_{n,k+1} = A_n\Big(\sum_{i=n+1}^{m}[\rho_i + \tau_{i,k}] + \sum_{i=1}^{n-1}[\rho_i + \tau_{i,k+1}] + \rho_n\Big). \tag{4.8}$$

Setting $\alpha_n \triangleq \frac{B_n - A_n}{A_n}$ and $\rho_\Xi \triangleq \sum_{i=1}^{m}\rho_i$, the above relationship can be simplified as

$$-\sum_{i=1}^{n-1}\tau_{i,k+1} + \alpha_n\tau_{n,k+1} = \rho_\Xi + \sum_{i=n+1}^{m}\tau_{i,k}. \tag{4.9}$$

Note that (4.9) can be written for all $n \in \Xi$ in a compact form using the vectors $\bar{\tau}_k = [\tau_{1,k}, \tau_{2,k}, \ldots, \tau_{m,k}]^T$, $\bar{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_m]^T$ and $\bar{1}_m = [1, 1, \ldots, 1]^T \in \mathbb{R}^m$, as:

$$\Delta_1\bar{\tau}_{k+1} = \Delta_2\bar{\tau}_k + \bar{1}_m\rho_\Xi, \tag{4.10}$$

where $\Delta_2 \in \mathbb{R}^{m\times m}$ is the strictly upper triangular matrix with all non-zero elements being 1 and $\Delta_1 = diag(\bar{\alpha}) - \Delta_2^T$. The affine linear system expression in (4.10) describes the evolution of agent dwell-times at targets on the target-cycle $\Xi$ over the number of tours completed $k$. To get an explicit expression for the steady state mean cycle uncertainty $J_{ss}(\Xi)$ defined in (4.7), the following three lemmas are used.

**Lemma 4.1.** *(Miller, 1981) Suppose $A \in \mathbb{R}^{m\times m}$ is an invertible matrix and $u, v \in \mathbb{R}^{m\times 1}$ are vectors. Then, $det(A + uv^T) = (1 + v^T A^{-1}u)det(A)$ and*

$$(1 + v^T A^{-1}u) \neq 0 \iff (A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}.$$

**Lemma 4.2.** *When $\sum_{i=1}^{m} \frac{A_i}{B_i} < 1$, the dynamic system given in (4.10) has a feasible equilibrium point $\bar{\tau}_{eq}$ (reached at $k = k_{eq}$),*

$$\bar{\tau}_{eq} = \left(\frac{\bar{\beta}}{1 - \bar{1}_m^T \bar{\beta}}\right)\rho_\Xi, \quad i.e., \ \tau_{n,k_{eq}} = \left(\frac{\beta_n}{1 - \sum_{i=1}^{m}\beta_i}\right)\rho_\Xi, \tag{4.11}$$

*for all $n \in \Xi$ with $\beta_n \triangleq \frac{A_n}{B_n}$ and $\bar{\beta} = [\beta_1, \beta_2, \ldots, \beta_m]^T$.*

*Proof.* See Appendix A.4.1. □

The following assumption is made to establish the stability of $\bar{\tau}_{eq}$ in (4.11).

**Assumption 4.3.** *The matrix $\Delta_1^{-1}\Delta_2$ is Schur stable (Bof et al., 2018).*

Based on several arguments, the work in (Welikala and Cassandras, 2019a) has conjectured that this assumption holds under some minor conditions. Nevertheless, since both $\Delta_1$ and $\Delta_2$ are known, this assumption's validity can be verified easily.

**Lemma 4.3.** *Under Assumption 4.3, the equilibrium point $\bar{\tau}_{eq}$ in (4.11) of the system (4.10) is globally asymptotically stable (i.e., $\lim_{k\to\infty} \bar{\tau}_k = \bar{\tau}_{eq}$, irrespective of $\bar{\tau}_0$).*

*Proof.* See Appendix A.4.2. □

**Theorem 4.1.** *Under Assumptions 4.2 and 4.3 with $\sum_{i=1}^{m} \frac{A_i}{B_i} < 1$, the generic (single-agent) unconstrained target-cycle $\Xi$ in Fig. 4·3 achieves a steady state mean cycle uncertainty value (i.e., (4.7)),*

$$J_{ss}(\Xi) = \frac{1}{2}(\bar{B} - \bar{A})^T \bar{\tau}_{eq}, \tag{4.12}$$

*where $\bar{B} = [B_1, B_2, \ldots, B_m]^T$, $\bar{A} = [A_1, A_2, \ldots, A_m]^T$, and $\bar{\tau}_{eq}$ is given in (4.11).*

*Proof.* See Appendix A.4.3. □

Theorem 4.1 enables assessing simple agent trajectories (e.g., Fig. 4·3) efficiently and will be used to construct a high-performing target-cycles on the graph $\mathcal{G}$.

### 4.2.2 Greedy Target-Cycle Construction

Theorem 4.1 can be used to identify the best performing (steady state, unconstrained) target-cycle in $\mathcal{C}$ if $|\mathcal{C}|$ is small via exhaustive search evaluating (4.12) over all $\Xi \in \mathcal{C}$:

$$\Xi^* = \underset{\Xi \in \mathcal{C}}{\arg\min}\ J_{ss}(\Xi). \tag{4.13}$$

Since this brute-force approach becomes computationally expensive as $|\mathcal{C}|$ grows exponentially with the number of targets or edges, an alternative computationally efficient greedy scheme is proposed to construct a sub-optimal target-cycle (denoted as $\Xi^{\#}$) as a candidate for $\Xi^*$ in (4.13). In this greedy scheme, each iteration search expands a current target-cycle $\Xi$ by adding an unvisited target $i \in \mathcal{V}\backslash\Xi$ to $\Xi$. The constructed $\Xi^{\#} \in \mathcal{C}$ is then transformed to a TCP which is used as $\Theta^{(0)}$ in (4.6). Therefore, determining the optimal target-cycle $\Xi^*$ is not essential at this stage as opposed to the importance of keeping the overall process of obtaining $\Theta^{(0)}$ efficient.

The *finite horizon mean cycle uncertainty* $J_T(\Xi_i)$ is now defined as

$$J_T(\Xi_i) \triangleq \frac{1}{T}\int_0^T \sum_{j \in \Xi_i} R_j(t)dt. \tag{4.14}$$

Note that if $\mathcal{V} = \Xi_i$, this $J_T(\Xi_i)$ metric is equivalent to main objective $J_T$ in (4.2).

**Contribution of a Neglected Target:** Formally, a *neglected target* is a target that is not visited by any agent during the period $[0, T]$. The following lemma characterizes the contribution of such a neglected target to the main objective $J_T$ in (4.2).

**Lemma 4.4.** *The contribution of a neglected target $i \in \mathcal{V}$ to the mean system uncertainty $J_T$ (defined in* (4.2)*) is* $\left(R_{i,0} + \frac{A_i T}{2}\right)$.

*Proof.* See Appendix A.4.4. □

**Assumption 4.4.** *For any target-cycle $\Xi \in \mathcal{C}$, the difference between the steady state mean cycle uncertainty* $J_{ss}(\Xi)$ (4.7) *and the finite horizon mean cycle uncertainty* $J_T(\Xi)$ (4.14) *is bounded by some finite constant $K_e \in \mathbb{R}_{\geq 0}$, i.e.,* $|J_{ss}(\Xi) - J_T(\Xi)| < K_e$.

The greedy target-cycle construction scheme uses the $J_{ss}(\cdot)$ metric defined in (4.7) to compare the performance of different target-cycles as it can be evaluated efficiently. However, since the original objective $J_T$ in (4.2) is evaluated over a finite horizon $T$, the $J_T(\cdot)$ metric defined in (4.14) is more appropriate to compare different target-cycle performances. The above assumption states that $J_T(\cdot)$ will always lie within $J_{ss}(\cdot) \pm K_e$. It is important to note that $K_e$ is small whenever: (i) the steady state tour duration $T_\Xi$ and the finite horizon $T$ is such that $T \gg T_\Xi$, and (ii) the dynamics of the steady state error of (4.10) are fast (i.e., from Lemma 4.3, when $\frac{A_i}{B_i} \ll 1$).

**Target-Cycle Expansion Operation (TCEO):** Consider the target-cycle $\Xi = \{1, 2, \ldots, m\}$ with its corresponding sequence of edges $\xi = \{(m, 1), (1, 2), \ldots, (m-1, m)\}$. As shown in Fig. 4·5, to expand $\Xi$ so that it includes one more target $i$ chosen from the set of neglected targets $\mathcal{V}\backslash\Xi$, (i) one edge $(n-1, n)$ chosen from $\xi$ should be replaced with two new consecutive edges $(n-1, i), (i, n) \in \mathcal{E}$ and (ii) the neglected target $i$ should be inserted into $\Xi$ between targets $n-1$ and $n$. Whenever $|\mathcal{V}\backslash\Xi| > 0$, the existence of a such $i$ and $(n-1, n)$ is guaranteed by the bi-triangularity condition in Assumption 4.1. Upon executing these two operations, a new (expanded) target-cycle $\Xi'$ (and $\xi'$) is attained as shown in Fig. 4·5. The following theorem derives the *marginal gain* (denoted as $\Delta J_T(i|\xi, (n-1, n))$) in the main objective $J_T$ in (4.2) due to such a target-cycle expansion in terms of $J_{ss}(\cdot)$ metric in (4.7).

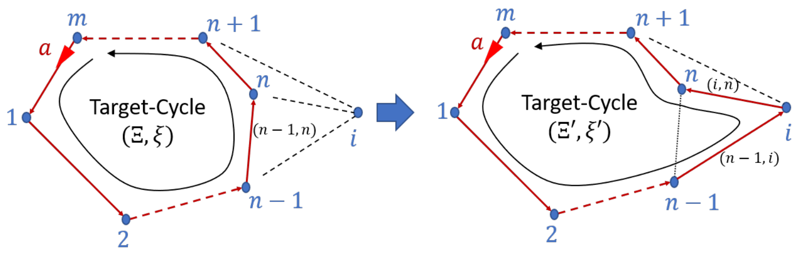


**Figure 4·5:** A basic target-cycle expanding operation (TCEO).

**Theorem 4.2.** *Under Assumptions 4.1, 4.2 and 4.4, the marginal gain in the main objective $J_T$ in (4.2) due to the target-cycle expansion operation shown in Fig. 4·5 is*

$$\Delta J_T(i|\xi,(n-1,n)) = \left(R_{i,0} + \frac{A_iT}{2}\right) + J_{ss}(\Xi) - J_{ss}(\Xi'), \tag{4.15}$$

*where $\Xi'$ is the expanded target-cycle and $J_{ss}(\cdot)$ is given in Theorem 4.1. The associated estimation error of this term is $\pm 2K_e$.*

*Proof.* See Appendix A.4.5. $\square$

**Greedy Algorithm:** Based on the discussion above and exploiting Theorem 4.2, Alg. 4.2 provides a systematic way to construct a candidate (sub-optimal) unconstrained target-cycle $\Xi^{\#}$ as a solution for (4.13). As described in (Welikala and Cassandras, 2019a), the resulting target-cycle $\Xi^{\#}$ can even be further refined using 2-Opt and 3-Opt local search techniques (Nilsson, 2003; Blazinskas and Misevicius, 2011).

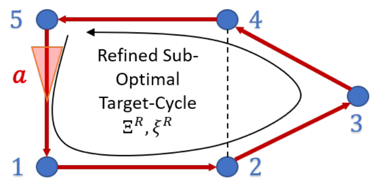
**Algorithm 4.2** Greedy target-cycle construction for (4.13).

```
   Input: Graph topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
   Output: A sub-optimal target-cycle $\Xi^{\#}$ (and $\xi^{\#}$) for (4.13).
1: Find the target-cycle $\Xi$ in $\mathcal{G}$ such that $|\Xi| = 2$ that has the minimum $J_{ss}(\Xi)$ value.
2: while True do
3:     Find the best way to expand $\Xi$ over all possible TCEOs.
4:     If its marginal gain $\Delta J_T > 0$, execute the expansion, otherwise, Break.
5: end while
6: $\Xi^{\#} := \Xi$; $\xi^{\#} := \xi$; Return;
```

### 4.2.3 Generating an Initial TCP: $\Theta^{(0)}$

Take the final refined sub-optimal target-cycle as $\Xi^R$ (and $\xi^R$). Now, $\Xi^R$ needs to be transformed into a set of TCP parameters to be used as $\Theta^{(0)}$ in (4.6). Since $\mathcal{A} = \{a\}$ under Assumption 4.1, $\Theta^{(0)} = \Theta^{a(0)} \in \mathbb{R}^{M\times M}$. Further, note that the TCP values in $\Theta^{(0)}$ should be such that they guide the agent according to Assumption 4.2 on $\Xi^R$. Algorithm 4.3 achieves this task as its Step 1 ensures that the agent remains at target $i \in \Xi^R$ until $R_i(t) = 0$ and Steps $2, 3$ ensure that the agent follows the target-cycle $\Xi^R$. An example input/output for this algorithm is shown in Fig. 4·6.

```
Algorithm 4.3 Generating $\Theta^{a(0)}$ from the target-cycle $\Xi^R, \xi^R$.
    Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and the target-cycle $\Xi^R, \xi^R$.
    Output: Initial TCP $\Theta^{a(0)}$ for the use in (4.6).
1: All the diagonal entries of $\Theta^{a(0)}$ are set to 0.
2: The $(i, j)^{\text{th}}$ entry of $\Theta^{a(0)}$ is set to 0 for all $(i, j) \in \xi^R$.
3: All other (valid/finite) entries of $\Theta^{a(0)}$ are set to a large constant $P \in \mathbb{R}$.
```



$$\Theta^{a(0)} = \begin{bmatrix} 0 & 0 & \infty & \infty & P \\ P & 0 & 0 & P & \infty \\ \infty & P & 0 & 0 & \infty \\ \infty & P & P & 0 & 0 \\ 0 & \infty & \infty & P & 0 \end{bmatrix}$$

**Figure 4·6:** The generated initial threshold matrix $\Theta^{a(0)}$ (right) for the refined sub-optimal target-cycle $\Xi^R$ (left).

## 4.3 Single-Agent PMN Solution - Part II

In this section, the bi-triangularity assumption in Assumption 4.1(ii) is relaxed so as to generalize the developed single-agent PMN solution for any network. In particular, this bi-triangularity assumption does not hold when the network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is sparse, due to the lack of edges in $\mathcal{E}$. As a result, the iterative target-cycle expansion process in Alg. 4.2 might halt prematurely (i.e., while $|\mathcal{V}\backslash\Xi| > 0$) due to the lack of feasible expansions. Two such examples are shown in Fig. 4·7. One obvious approach to overcome this assumption violation is by inserting (artificial) edges into the network with higher travel-time values. However, while such an approach can make Alg. 4.2 run without halting, the resulting target-cycle $\Xi^{\#}$ will contain the edges that were artificially introduced, compromising the target-cycle performance $J_{ss}(\Xi^{\#})$.

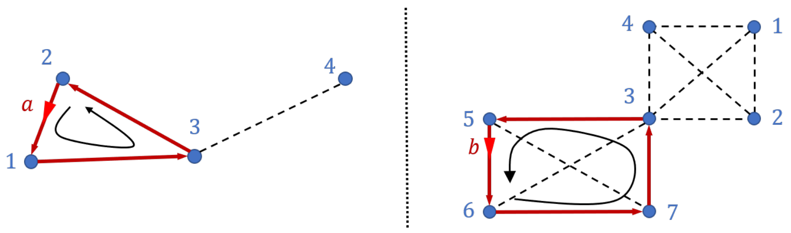


**Figure 4·7:** Two example sparse networks where Alg. 4.2 has halted prematurely while executing target-cycle expansion iterations.

**Auxiliary Targets:** As opposed to introducing artificial edges, this work proposes to introduce artificial targets (henceforth called *auxiliary targets*) into the network so as to deal with the issue of premature halting. Unlike artificial edges, an auxiliary target is always associated with a corresponding target in the original network and its exact physical interpretation is provided in the sequel.

Note that if certain targets in the network can be visited more than once, the target-cycle expansion process may not have to be halted due to the lack of edges (sparseness or non-bi-triangularity) in the network. Therefore, this work proposes to allow targets to be visited more than once during a tour on a target-cycle. Such target-cycles are called *constrained target-cycles*. For example, in both networks shown in Fig. 4·7, if target 3 is allowed to be visited more than once during a tour, the constrained target-cycles $\bar{\Xi} = \{2,1,3,4,3\}$ and $\bar{\Xi} = \{6,7,3,2,1,4,3,5\}$ could have been constructed, respectively. Note that the notation "$\bar{\cdot}$" is used to indicate that the target-cycle is constrained (i.e., some elements are being repeated).

To analyze such constrained target-cycles (to evaluate $J_{ss}(\cdot)$ in (4.7)), the said concept of *auxiliary targets* is used. The idea is to replace the repeated targets in the constrained target-cycle with a set of carefully chosen auxiliary targets to create an *equivalent* unconstrained target-cycle enabling the application of Theorem 4.1.

Consider a constrained target-cycle $\bar{\Xi}$ with a target $i \in \bar{\Xi}$ being visited $n$ times during a tour. First, an *auxiliary target pool* $\mathcal{T}_i = \{i^1, i^2, \ldots, i^n\}$ is introduced where each auxiliary target $i^j \in \mathcal{T}_i$ can be thought of as an artificial target located at the same physical location of target $i$ (i.e., at $Y_i$), but with its own parameters: an uncertainty rate $A_i^j$ and a sensing rate $B_i^j$ (to be defined). Next, the repeated elements of target $i$ in $\bar{\Xi}$ are replaced with the elements taken from $\mathcal{T}_i$ and the process is repeated for all $i \in \bar{\Xi}$ with $|\mathcal{T}_i| > 1$. This results in an unconstrained target-cycle $\Xi$ (i.e., without "$\bar{\cdot}$", this notational convention is followed in the sequel). Figure 4·8 shows two such unconstrained target-cycles obtained from introducing auxiliary targets to transform respective constrained target-cycles proposed for Fig. 4·7.

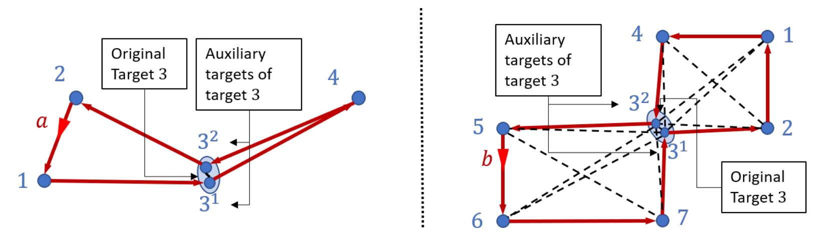


**Figure 4·8:** Converting constrained target-cycles into unconstrained target-cycles with the use of auxiliary targets.

**Equivalence Criteria:** For the analysis of the constrained target-cycles, it is enforced that both the targets in $\Xi$ and $\bar{\Xi}$ should perform/behave in an equivalent manner at steady state. In particular, the following *equivalence criteria* is enforced:

1. The dwell-time spent at $i^j \in \Xi$ is equal to the dwell-time spent at $i \in \bar{\Xi}$ on its $j^{\text{th}}$ visit during a tour.

2. The physical location of $i^j \in \Xi$ is the same as that of $i \in \bar{\Xi}$.

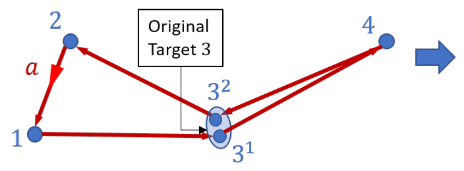
3. The total contribution to the main objective $J_T$ (4.2) by $\mathcal{T}_i \subset \Xi$ is equal to that of target $i \in \bar{\Xi}$, during a tour.

The first two conditions ensure that the tour duration is the same for both $\Xi$ and $\bar{\Xi}$. The third condition implies $J_{ss}(\bar{\Xi}) = J_{ss}(\Xi)$. Hence, if the auxiliary target parameters are known, $J_{ss}(\bar{\Xi})$ can be evaluated using Theorem 4.1.

**Sub-Cycles:** Each $i^j \in \Xi$ can be assigned a *sub-cycle* denoted by $\Xi_i^j \subset \Xi$ where $\Xi_i^j$ starts with the immediate next target to $i^{j-1} \in \Xi$ and ends with target $i^j$. Therefore, $\Xi$ can be written as a concatenation of sub-cycles of a target $i \in \bar{\Xi}$, i.e., $\Xi = \bigcup_{i^j \in \mathcal{T}_i} \Xi_i^j$. For example, for the unconstrained target-cycle $\Xi$ shown in Fig. 4·8(left), sub-cycles corresponding to $3^1, 3^2 \in \Xi$ are $\Xi_3^1 = \{2, 1, 3^1\}$ and $\Xi_3^2 = \{4, 3^2\}$, respectively.

The *sub-cycle unit vector* of $\Xi_i^j$ is denoted by $\bar{1}_i^j \in \mathbb{R}^{|\Xi|}$ and its $n^{\text{th}}$ element is 1 only if the $n^{\text{th}}$ element of $\Xi$ belongs to $\Xi_i^j$. Therefore, if $\bar{1}_{|\Xi|} \in \mathbb{R}^{|\Xi|}$ is a vector of all ones, with respect to target $i \in \bar{\Xi}$, $\bar{1}_{|\Xi|} = \sum_{i^j \in \mathcal{T}_i} \bar{1}_i^j$.

The *sub-cycle matrix* of $\Xi$ is denoted by $\mathbf{1}_\Xi \in \mathbb{R}^{|\Xi| \times |\Xi|}$ and its $n^{\text{th}}$ column is the sub-cycle unit vector of the $n^{\text{th}}$ element of $\Xi$. An example is shown in Fig. 4·9.



$$\bar{\Xi} = \{1,\ 3,\ 4,\ 3,\ 2\}$$
$$\Xi = \{1,\ 3^1,\ 4,\ 3^2,\ 2\}$$
$$\mathbf{1}_\Xi = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

**Figure 4·9:** Sub-cycle unit vectors and sub-cycle matrix (right) for a given constrained target-cycle $\bar{\Xi}$ (left).

### 4.3.1 Analysis of Constrained Target-Cycles

A generic constrained target-cycle $\bar{\Xi}$ is analyzed in this section. For illustrative purposes the constrained target-cycle example shown in Fig. 4·10 is used. In there, note that $\bar{\Xi} = \{1, 2, \ldots, n, \ldots, n+m-1, n\}$ and target $n \in \bar{\Xi}$ is visited twice during a tour. Introducing auxiliary targets $\mathcal{T}_n = \{n^1, n^2\}$, $\bar{\Xi}$ can be converted to its equivalent unconstrained version $\Xi$. The sub-cycles of $n^1$ and $n^2$ in $\Xi$ are $\Xi_n^1 = \{1, 2, \ldots, n-1, n^1\}$ and $\Xi_n^2 = \{n+1, n+2, \ldots, n+m-1, n^2\}$, respectively. A *tour* on $\bar{\Xi}$ starts/ends when the agent leaves target $n$ to reach target 1 and the agent behavior on $\bar{\Xi}$ is assumed to follow Assumption 4.2. The inter-target travel-times on $\bar{\Xi}$ are labeled similar to before (see Figs. 4·3) and $\bar{\rho}_{\Xi} = [\rho_1, \rho_2, \ldots, \rho_n^1, \ldots, \rho_{n+m-1}, \rho_n^2]^T$ is used to denote the *travel-time vector* of $\bar{\Xi}$. The transient analysis of the constrained target-cycle $\bar{\Xi}$ is omitted here by making the following assumption (see Remark 4.1).

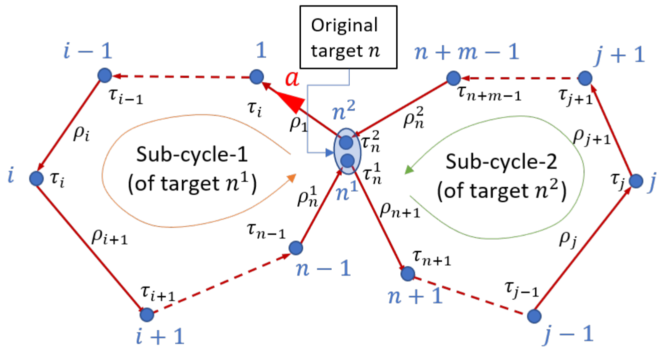


**Figure 4·10:** A general constrained target-cycle with target $n$ being visited twice during the cycle.

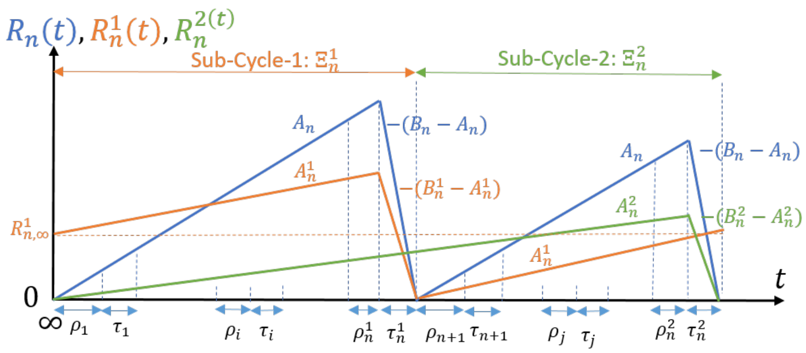


**Figure 4·11:** Variation of the target uncertainties of the constrained target-cycle shown in Fig. 4·10 - after achieving steady state.

**Assumption 4.5.** *The dwell-time dynamics of the constrained target-cycle $\bar{\Xi}$ have a feasible and globally asymptotically stable equilibrium point.*

Figure 4·11 shows the steady state behavior of the target uncertainties during a tour on the target-cycle $\bar{\Xi}$. The notation $\bar{\tau}_\Xi = [\tau_1, \tau_2, \ldots, \tau_n^1, \ldots, \tau_{n+m-1}, \tau_n^2]^T$ is used to represent the steady state dwell-times of targets in $\Xi$. The following lemma generalizes Lemma 4.2 to evaluate $\bar{\tau}_\Xi$ for any target-cycle $\bar{\Xi}$.

**Lemma 4.5.** *Under Assumptions 4.2 and 4.5, when a single agent traverses a generic constrained target-cycle $\bar{\Xi}$ (with $\Xi$ being the equivalent unconstrained version of $\bar{\Xi}$), the steady state dwell-times $\bar{\tau}_\Xi$ are given by*

$$\bar{\tau}_\Xi = [diag(\bar{\gamma}_\Xi) - \mathbf{1}_\Xi]^{-1}\mathbf{1}_\Xi\bar{\rho}_\Xi, \tag{4.16}$$

*where $\bar{\gamma}_\Xi \in \mathbb{R}^{|\Xi|}$ is such that if the $i^{th}$ target of $\bar{\Xi}$ is $j$, then, the $i^{th}$ element of $\bar{\gamma}_\Xi$ is $\frac{B_j}{A_j}$ and $\mathbf{1}_\Xi$ is the sub-cycle matrix and $\bar{\rho}_\Xi$ is the travel-time vector of $\Xi$.*

*Proof.* See Appendix A.4.6. □

**Remark 4.1.** *Note that* (4.16) *is only valid under Assumption 4.5, i.e., if the dwell-times observed in the $k^{th}$ tour on $\bar{\Xi}$ (say $\bar{\tau}_{\Xi,k}$) converge to an equilibrium point ($\bar{\tau}_\Xi$) as $k \to \infty$. However, based on the form of* (4.16), *it can be concluded that the conditions for the existence and feasibility of such an equilibrium point are $|diag(\bar{\gamma}_\Xi) - \mathbf{1}_\Xi| \neq 0$ and $[diag(\bar{\gamma}_\Xi) - \mathbf{1}_\Xi]^{-1}\mathbf{1}_\Xi\bar{\rho}_\Xi > 0$, respectively.*

Using the dwell-time vector $\bar{\tau}_\Xi$ given by Lemma 4.5, the *total sub-cycle time* denoted by $T_n^j$ can be determined for all $n^j \in \Xi$ using $T_n^j = (\bar{1}_n^j)^T(\bar{\rho}_\Xi + \bar{\tau}_\Xi)$. Moreover, the *total cycle time* denoted by $T_\Xi$ can be determined using $T_\Xi = \bar{1}_{|\Xi|}^T(\bar{\rho}_\Xi + \bar{\tau}_\Xi)$.

**Lemma 4.6.** *Under the same conditions stated in Lemma 4.5, the auxiliary target parameters of any $n^j \in \Xi$ (i.e., $A_n^j$ and $B_n^j$) are:*

$$A_n^j = \frac{T_n^j}{T_\Xi}\frac{\tau_n^j(B_n - A_n)}{(T_\Xi - \tau_n^j)} \quad \textit{and} \quad B_n^j = \frac{T_n^j(B_n - A_n)}{(T_\Xi - \tau_n^j)}. \tag{4.17}$$

*Proof.* See Appendix A.4.7. □

With the auxiliary target parameters given by Lemma 4.6, all the respective parameters of targets in $\Xi$ can be lumped into vectors as $\bar{A}_\Xi$ and $\bar{B}_\Xi$. For example, for the target-cycle shown in Fig. 4·10, $\bar{A}_\Xi = [A_1, A_2, \ldots, A_n^1, \ldots, A_{n+m-1}, A_n^2]^T$.

**Theorem 4.3.** *Under Assumptions 4.2 and 4.5, when a single agent traverses a generic constrained target-cycle $\bar{\Xi}$ (with $\Xi$ being the equivalent unconstrained version of $\bar{\Xi}$), the steady state mean cycle uncertainty $J_{ss}(\bar{\Xi})$ (defined in (4.7)) is*

$$J_{ss}(\bar{\Xi}) = \frac{1}{2}(\bar{B}_\Xi - \bar{A}_\Xi)^T \bar{\tau}_\Xi, \tag{4.18}$$

*where the steady state dwell-times vector $\bar{\tau}_\Xi$ is given by Lemma 4.5 and the auxiliary target parameters included in the vectors $\bar{A}_\Xi$ and $\bar{B}_\Xi$ are given by Lemma 4.6.*

*Proof.* Since $\Xi$ is an unconstrained target-cycle, Theorem 4.1 gives $J_{ss}(\Xi) = \frac{1}{2}(\bar{B}_\Xi - \bar{A}_\Xi)^T \bar{\tau}_\Xi$, where $\bar{\tau}_\Xi$ is given by Lemma 4.5 and unknown parameters in $\bar{A}_\Xi$ and $\bar{B}_\Xi$ are given by Lemma 4.6. Finally, due to the equivalence criterion 3: $J_{ss}(\bar{\Xi}) = J_{ss}(\Xi)$. □

### 4.3.2 Greedy Target-Cycle Construction

Let $\mathcal{D}$ denote the set of all possible target-cycles on $\mathcal{G}$. Clearly, $\mathcal{D} \supseteq \mathcal{C}$ and $|\mathcal{D}| = \infty$ (see also (4.13)). Thus, exhaustive search methods (that exploit Theorem 4.3) cannot be used to determine the best performing (at steady state) target-cycle in $\mathcal{D}$:

$$\bar{\Xi}^* = \underset{\bar{\Xi} \in \mathcal{D}}{\arg\min}\ J_{ss}(\bar{\Xi}). \tag{4.19}$$

Hence, an efficient greedy scheme (identical to Alg. 4.2) is proposed to construct a sub-optimal target-cycle $\bar{\Xi}^{\#} \in \mathcal{D}$ as a candidate for $\bar{\Xi}^*$ in (4.19).

Notice that having the capability to make repeated visits to the targets during a tour on a target-cycle provides flexibility in ways in which a given target-cycle can be expanded. Hence, in this paradigm, apart from the previously used basic target-cycle expansion operation (labeled TCEO-1 and shown in Fig. 4·12(b)), two new TCEOs (labeled TCEO-2, TCEO-3 shown respectively in Fig. 4·12(c), (d)) are also proposed. Details are omitted here but can be found in (Welikala and Cassandras, 2019a).

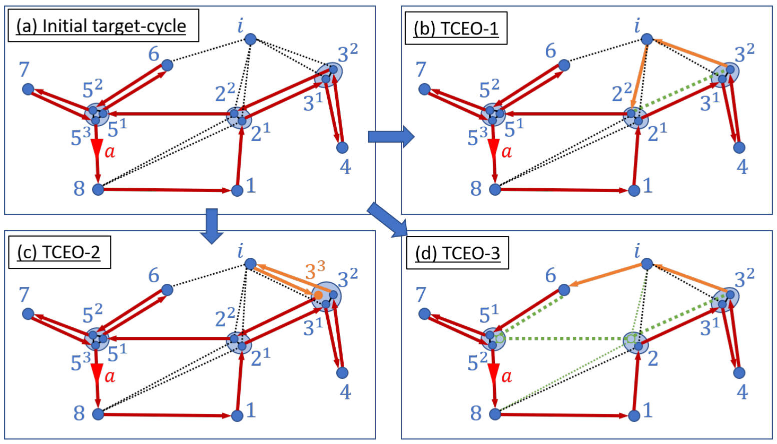


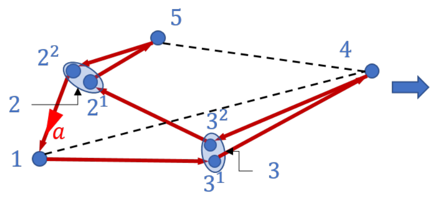
**Figure 4·12:** Target-cycle expansion operations (TCEOs).

Regardless of the type of the TCEO, note that Theorems 4.3 and 4.2 can be used to determine the corresponding marginal gain. Therefore, an identical target-cycle construction algorithm to the one shown in Alg. 4.2 can be used for the purpose of constructing a sub-optimal target-cycle $\bar{\Xi}^{\#} \in \mathcal{D}$ as a candidate for $\bar{\Xi}^{*}$ in (4.19). However, note that in each greedy target-cycle expansion iteration (i.e., in Step 3 of Alg. 4.2), the best feasible target-cycle expansion considering all three types of TCEOs (not limiting to TCEO-1) should be determined. These additional two greedy search space dimensions introduced (due to TCEO-2 and TCEO-3) resolve the issue of 'premature halting' of the target-cycle expansion process, as there is always a feasible target-cycle expansion (of type TCEO-2) whenever there are neglected targets (i.e., when $|\mathcal{V}\backslash\bar{\Xi}| > 0$) - given the network is connected. Moreover, the TSP inspired target-cycle refinements (based on 2-Opt and 3-Opt local search techniques (Blazinskas and Misevicius, 2011)) are also applicable here to obtain a refined target-cycle denoted $\bar{\Xi}^{R}$ from $\bar{\Xi}^{\#}$.

### 4.3.3 Generating an Initial TCP: $\Theta^{(0)}$

Recall that $\bar{\Xi}^{R}$ should be transformed into a set of TCP parameters to be used as $\Theta^{(0)}$ in (4.6). Due to the single-agent assumption (i.e., Assumption 4.1(i)), $\Theta^{(0)} = \Theta^{a(0)}$.

Note that even though $\bar{\Xi}^R$ might be a constrained target-cycle, Alg. 4.3 can still be used to get the corresponding TCP $\Theta^{a(0)}$, but under few minor conditions (provided in (Welikala and Cassandras, 2019a)). Figure 4·13 shows an example constrained target-cycle and its corresponding TCP $\Theta^{a(0)}$ given by Alg. 4.3.



$$\bar{\Xi}^{\mathrm{R}} = \{1,3,4,3,2,5,2\}$$
$$\Xi^{R} = \{1,3^1,4,3^2,2^1,5,2^2\}$$
$$\Theta^{a(0)} = \begin{bmatrix} 0 & P & 0 & P & \infty \\ 0 & 0 & P & \infty & 0 \\ P & 0 & 0 & 0 & \infty \\ P & \infty & 0 & 0 & P \\ \infty & 0 & \infty & P & 0 \end{bmatrix}$$

**Figure 4·13:** The generated threshold matrix $\Theta^{a(0)}$ for the refined sub-optimal target-cycle $\bar{\Xi}^R$ shown (left).

## 4.4 Multi-Agent PMN Solution

The previous two sections focused on the single-agent PMN problem and developed techniques to (i) identify a favorable agent trajectory in a given network and (ii) transform the identified trajectory into a TCP $\Theta^{(0)}$ for the subsequent use in a gradient process (4.6). To conveniently generalize these single-agent techniques to multi-agent PMNs, as outlined in Alg. 4.1, the network $\mathcal{G}$ is proposed to be partitioned into $N$ sub-graphs (recall $N = |\mathcal{A}|$). This 'divide and conquer' approach enables the use of developed single-agent techniques (behind Steps 2 and 5 of Alg. 4.1) independently in each of the sub-graphs. Therefore, this section presents the proposed graph partitioning, refining and agent assigning processes that respectively correspond to Steps 1, 3 and 4 of Alg. 4.1. For these processes, several known techniques from (von Luxburg, 2007; Ng et al., 2001; Jianbo Shi and Malik, 2000; Ahuja et al., 1993) are adopted. Thus, some technical details are omitted (but provided in (Welikala and Cassandras, 2019a)) to emphasize the contributions of this thesis to achieve such an adaptation.

### 4.4.1 Graph Partitioning via Spectral Clustering

To partition the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, this thesis proposes to use *spectral clustering* (von Luxburg, 2007) - which is a commonly used global graph partitioning method that also has the advantages of: (i) simple implementation, (ii) efficient evaluation and (iii) better results compared to traditional techniques such as the $k$-means algorithm (von Luxburg, 2007). In spectral clustering, the graph partitions of $\mathcal{G}$ are derived based on a set of inter-target *similarity* values $\{s_{ij} : i, j \in \mathcal{V}\}$ so that the similarity value between two targets is high if they belong to the same partition and low otherwise.

**Remark 4.2.** *In a typical data-point clustering application, the graph representation (also called the "similarity graph") arises from the known similarity values between the data-points. However, in PMN problems, while the physical graph $\mathcal{G}$ is known, the similarity values between its targets (hence the similarity graph) are unknown.*

**Deriving Similarity Values:** In this work, the knowledge of the target topology $\mathcal{G}$ and target parameters are exploited to derive appropriate similarity values. Typically, a similarity value $s_{ij} \geq 0$ is obtained based on a *disparity* value $d(i, j)$ as

$$s_{ij} = \exp\left(-\frac{|d(i,j)|^2}{2\sigma^2}\right), \quad i, j \in \mathcal{V}, \tag{4.20}$$

where $d : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ and $\sigma^2$ is a user defined scaling parameter that controls how rapidly the similarity $s_{ij}$ falls off with disparity $d(i, j)$ (Ng et al., 2001). This function (4.20) is known as the *Gaussian similarity function.*

For the considered PMN problem setup, neither of using $d(i, j)$ as the *physical distance* (i.e., $\|Y_i - Y_j\|$) nor the *shortest distance* between the targets $i$ and $j$ provides a good characterization to the underlying persistent monitoring aspects of the problem as they disregard target parameters and agent behaviors when monitoring targets.

Therefore, to obtain similarity values, a novel disparity metric named *minimum* *mean covering cycle uncertainty* (CCU) is proposed as

$$d(i,j) = d_{CC}(i,j) \triangleq \min_{\bar{\Xi}:\, i,j \in \bar{\Xi}} J_{ss}(\bar{\Xi}). \tag{4.21}$$

The arg min of the above problem is named the *optimal covering cycle* (OCC) and is denoted as $\bar{\Xi}_{ij}^*$. Simply, the OCC $\bar{\Xi}_{ij}^*$ is the best way to cover targets $i$ and $j$ in a single target-cycle so that the corresponding $J_{ss}(\cdot)$ value is minimized. Therefore, if the CCU value is higher for a certain target pair, it implies that it is difficult to cover those two targets in a single target-cycle. Hence, it is clear that this disparity metric $d_{CC}(i,j)$ in (4.21) provides a good characterization to the underlying persistent monitoring aspects of the PMN problems.

To estimate $d_{CC}(i,j),\ \forall i,j \in \mathcal{V}$, a modified version of the Dijkstra's algorithm (Ahuja et al., 1993) coupled with cycle expanding and refining techniques discussed in Section 4.3 is used (details can be found in (Welikala and Cassandras, 2019a)). Subsequently, (4.20) is used to obtain the the respective similarity values $s_{ij}\ \forall i,j \in \mathcal{V}$.

**Spectral Clustering Algorithm:** Finally, based on the obtained similarity values, the normalized spectral clustering (Jianbo Shi and Malik, 2000) is applied to derive the set of target partitions of $\mathcal{V}$, i.e., $\{\mathcal{V}_a : a \in \mathcal{A}\}$ and the respective sub-graphs $\{\mathcal{G}_a : a \in \mathcal{A}\}$ where each $\mathcal{G}_a = (\mathcal{V}_a, \mathcal{E}_a)$ and $\mathcal{E}_a \subseteq \mathcal{E}$ is the set of intra-cluster edges.

### 4.4.2 Refining the Graph Partitions

Once the sub-graphs are formed (Step 1 of Alg. 4.1), the target-cycle construction process (Step 2 of Alg. 4.1) is executed on each sub-graph. The resulting target-cycle on a sub-graph $\mathcal{G}_a$ is denoted as $\bar{\Xi}_a$ and is assumed to be assigned to an arbitrary agent $a \in \mathcal{A}$ (in Section 4.4.3, target-cycles will be explicitly assigned to the agents).

An agent $a \in \mathcal{A}$ can remove a target $i \in \bar{\Xi}_a$ from its target-cycle $\bar{\Xi}_a$ by reconstructing a new target-cycle over its sub-graph $\mathcal{G}_a$ while ignoring target $i$. Such a process is called as a target-cycle *contraction*. In contrast, an agent $b \in \mathcal{A}$ can expand its target-cycle $\bar{\Xi}_b$ to include an external target $i \notin \bar{\Xi}_b$ by simply carrying out the best possible target-cycle *expansion* out of the three TCEOs shown in Fig. 4·12. Using such contraction and expansion operations, two agents $a,\, b \in \mathcal{A}$ can *trade* a target $i \in \bar{\Xi}_a$ between each other (i.e., between sub-graphs $\mathcal{G}_a$ and $\mathcal{G}_b$), if the *marginal gain* $\Delta J_{ss}^{ab,i} \triangleq (J_{ss}(\bar{\Xi}_a) + J_{ss}(\bar{\Xi}_b)) - (J_{ss}(\bar{\Xi}_a') + J_{ss}(\bar{\Xi}_b')) > 0$, where $\bar{\Xi}_a'$ and $\bar{\Xi}_b'$ are the contracted and expanded target-cycles, respectively.

A set of sub-graphs is called “balanced” if there is no $a, b \in \mathcal{A}$ and $i \in \mathcal{V}_a$ such that $\Delta J_{ss}^{ab,i} > 0$. The spectral clustering method often provides a balanced set of sub-graphs. Nevertheless, a distributed greedy algorithm is proposed for the agents to balance the sub-graphs by systematically executing trades with positive marginal gains (details are provided in (Welikala and Cassandras, 2019a)). The convergence of a such greedy algorithm is guaranteed as each greedy step (i.e., each “trade”) decreases the metric: $\sum_{a\in\mathcal{A}} J_{ss}(\bar{\Xi}_a)$, which is lower bounded by 0.

### 4.4.3 Assigning Agents to the Target-Cycles

So far, a set of target-cycles $\{\bar{\Xi}_b : b \in \mathcal{B}\}$ has been identified on a respective set of (balanced) sub-graphs of $\mathcal{G}$, where $\mathcal{B}$ is the set of target-cycle indexes ($\mathcal{B} \equiv \mathcal{A}$). These target-cycles $\{\bar{\Xi}_b : b \in \mathcal{B}\}$ are now explicitly assigned to the agents based on initial agent locations $\{s_a(0) : a \in \mathcal{A}\}$. First, the assignment cost between an agent $a \in \mathcal{A}$ and a target-cycle $\bar{\Xi}_b$, $b \in \mathcal{B}$ is defined as $h_{ab}$ where $h_{ab}$ represents the total travel-time on the fastest available path to reach any one of the targets in $\bar{\Xi}_b$ starting from $s_a(0)$. Then, Dijkstra’s shortest path algorithm (Ahuja et al., 1993) is used to compute all these assignment weights. Subsequently, the assignment problem (between $a$’s and $b$’s) is solved using the shortest augmenting path algorithm (Ahuja et al., 1993).

**Generating an Initial TCP:** $\Theta^{(0)}$ Assume an agent $a \in \mathcal{A}$ is optimally assigned to the target-cycle $\bar{\Xi}_b$ and the corresponding fastest path from $s_a(0)$ to reach $\bar{\Xi}_b$ is $\Phi_{ab} = \{i_1, i_2, \ldots, i_n\} \subset \mathcal{V}$. Note that $i_n \in \bar{\Xi}_b$, $Y_{i_1} = s_a(0)$ and take $\Phi'_{ab} = \Phi_{ab} \backslash \{i_n\}$. Now, Alg. 4.3 can be used with $\bar{\Xi}_b$ to get a corresponding TCP for agent $a$ as $\Theta^a$. Note that this only assigns the set of rows: $\{j : j \in \bar{\Xi}_b\}$ in $\Theta^a$ as it is sufficient to keep the agent on the target-cycle $\bar{\Xi}_b$ (this corresponds to rows 1-3 in the example TCP $\Theta^a$ shown in Fig. 4·14). Therefore, to make sure that the agent $a$ follows the path $\Phi_{ab}$, the set of rows: $\{j : j \in \Phi'_{ab}\}$ in $\Theta^a$ are assigned as follows (this corresponds to rows 4-5 in the example TCP $\Theta^a$ shown in Fig. 4·14). If $j$ and $k$ are two consecutive entries in $\Phi_{ab}$, in the $j^{\text{th}}$ row of $\Theta^a$ set: $\theta^a_{jj} = 0, \theta^a_{jk} = 0$ and any other entry $\theta^a_{jl}$ to $P$ or $\infty$ depending on whether $(j,l) \in \mathcal{E}$. Finally, set $\Theta^{a(0)} = \Theta^a$ for the use in (4.6).

With this, all the steps involved in Alg. 4.1, i.e., the proposed PMN solution in this chapter, now have been covered.

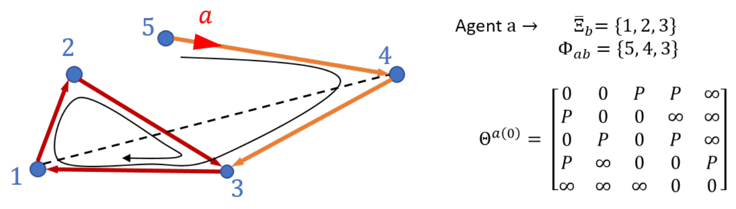

5
$a$
2
4
1
3
Agent a → $\bar{\Xi}_b = \{1,2,3\}$
$\Phi_{ab} = \{5,4,3\}$
$\Theta^{a(0)} = \begin{bmatrix} 0 & 0 & P & P & \infty \\ P & 0 & 0 & \infty & \infty \\ 0 & P & 0 & P & \infty \\ P & \infty & 0 & 0 & P \\ \infty & \infty & \infty & 0 & 0 \end{bmatrix}$

**Figure 4·14:** The generated initial TCP $\Theta^{a(0)}$ when the agent $a$ is initially at target 5 and have been assigned to the target-cycle $\bar{\Xi}_b = \{3, 1, 2\}$ with the fastest path being $\Phi_{ab} = \{5, 4, 3\}$.

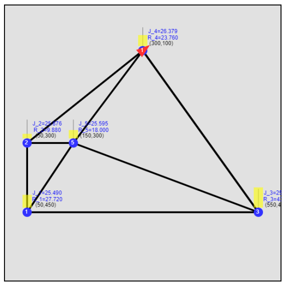
## 4.5 Simulation Results

In this section, several numerical examples are provided to show how the proposed greedy initialization process can benefit the performance of the TCP used in solving PMN problems. First, consider the single-agent PMN problem configuration (labeled SASE1) shown in Fig. 4·15(a). In this figure (and similar figures used in the sequel), blue circles represent the targets, while black lines represent available trajectory segments that agents can take to travel between targets. Red triangles and the yellow vertical bars respectively indicate the agent locations and the target uncertainty levels. Since both of those quantities are time-varying (i.e., $s_a(t)$ and $R_i(t)$), their terminal state is indicated (i.e., at $t = T$, when the best TCP found so far is used). In all numerical examples, the PMN problem parameters have been chosen as follows. The target parameters are: $A_i = 1,\ B_i = 10,\ R_i(0) = 0.5,\ \forall i \in \mathcal{V}$ and all the targets have been placed inside a $600 \times 600$ mission space. The time horizon is taken as $T = 500$. Each agent is assumed to have first-order dynamics (following from (Zhou et al., 2019)) with a maximum speed of 50 units per second. The initial locations of the agents are chosen such that they are uniformly distributed among the targets at $t = 0$ (i.e., $s_a(0) = Y_i$ with $i = 1 + (a-1) * \text{round}(M/N)$).

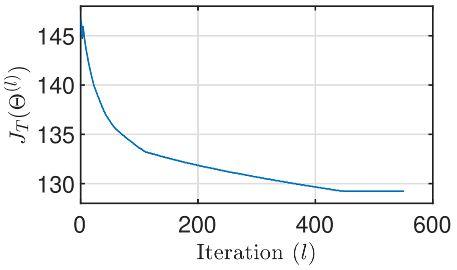
The proposed PMN solution in this chapter (i.e., Alg. 4.1) including the method proposed in (Zhou et al., 2019) have been implemented in a JavaScript-based interactive simulation platform available at `http://www.bu.edu/codes/simulations/shiran27/PersistentMonitoring/`. Readers are invited to reproduce the reported results and also to try new problem configurations using this simulator.

Figure 4·15(b) shows the corresponding evolution of $J_T(\Theta^{(l)})$ when $\Theta^{(l)}$ was updated according to (4.6) starting from a randomly selected $\Theta^{(0)}$ (Zhou et al., 2019).

Next, the PMN solution proposed in this chapter is applied to the SASE1. First, a high-performing target-cycle was constructed using the proposed Alg. 4.2. Figures 4·16(a)$\rightarrow$(d) show the intermediate target-cycles observed (as red traces) during this process. The resulting target-cycle in Fig. 4·16(d) is $\bar{\Xi}^R = \{2, 1, 2, 5, 3, 4, 5\}$ with $J_{ss}(\bar{\Xi}^R) = 121.1$. Then, $\bar{\Xi}^R$ was transformed into a TCP $\Theta^{(0)}$ using Alg. 4.3 to initialize the gradient descent (4.6). Figure 4·17(b) shows that the obtained TCP $\Theta^{(0)}$ results in a $J_T$ value of 114.9 which is not further improved from the gradient descent (4.6), as $\Theta^{(0)}$ is directly locally optimal. Note however that this solution is 11.1% better than the solution given by (Zhou et al., 2019) (shown in Fig 4·15).

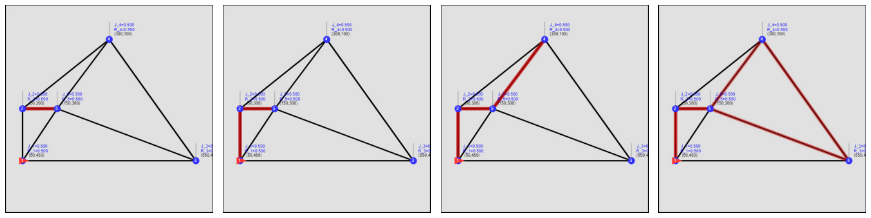


**(a)** Config. at $t = T$.



**(b)** Cost vs iterations plot.

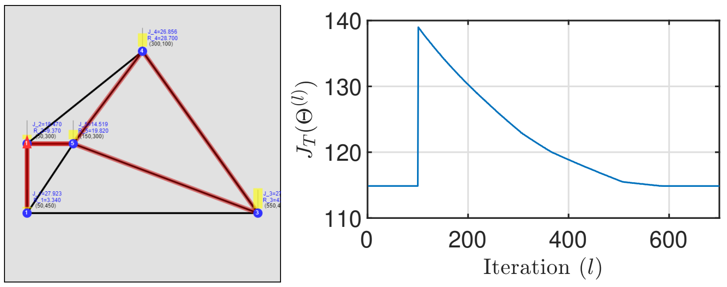
**Figure 4·15:** Single-agent simulation example 1 (SASE1): Started with a random $\Theta^{(0)}$, converged to a TCP with $J_T = 129.2$.



**(a)** Iter. 1 **(b)** Iter. 2 **(c)** Iter. 3 **(d)** Iter. 3: $\bar{\Xi}^R$

**Figure 4·16:** Greedy target-cycle construction for the SASE1.

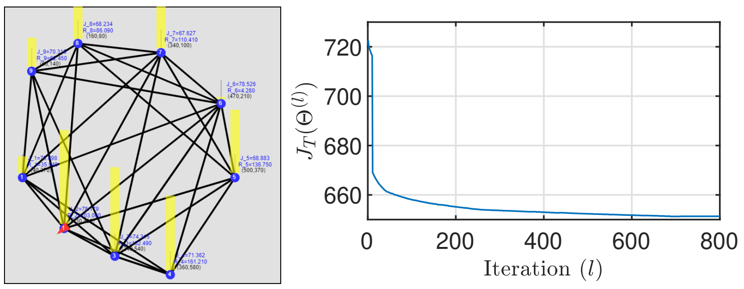
The second simulation example shown in Fig. 4·18(a) (labeled SASE2) aims to highlight the importance of gradient steps (4.6). As shown in Fig. 4·18(b), when $\Theta^{(0)}$ was selected randomly, (4.6) converges to a TCP with $J_T = 651.3$. In contrast, when $\Theta^{(0)}$ was derived using the greedy method proposed in this chapter, it directly yields $J_T = 607.9$. Next, when (4.6) was used to further update this TCP, unlike in SASE1, an improvement in $J_T$ was observed, finally reaching $J_T = 567.0$ (see Fig. 4·19(b) and (c)). The main difference between the solutions in Fig. 4·19(a) and (b) is that in the former the agent avoids visiting target 4, whereas in the latter, the agent visits target 4. Compared to (Zhou et al., 2019), the percentage improvement achieved from deploying the proposed PMN solution is 12.9%. A similar simulation



**(a)** Config. at $t = T$. **(b)** Cost vs iterations plot.

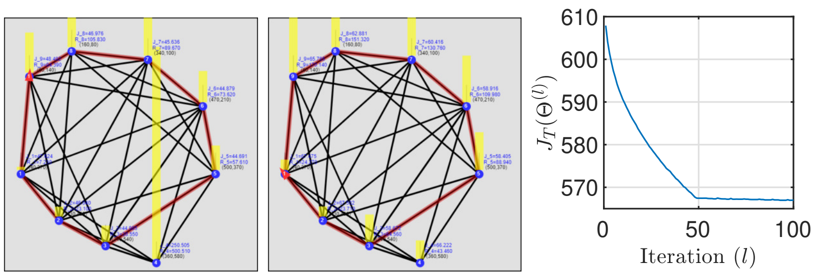
**Figure 4·17:** SASE1: The TCP $\Theta^{(0)}$ given by the target-cycle $\bar{\Xi}^R$ (the red trace in (a)) shows local optimality. At $l = 100, \Theta^{(l)}$ is randomly perturbed. Yet, converges back to the initial TCP. Cost $J_T = 114.9$ (Improvement $= +14.3$ compared to Fig. 4·15).

example (labeled SASE3) is shown in Figs. 4·20 and 4·21.



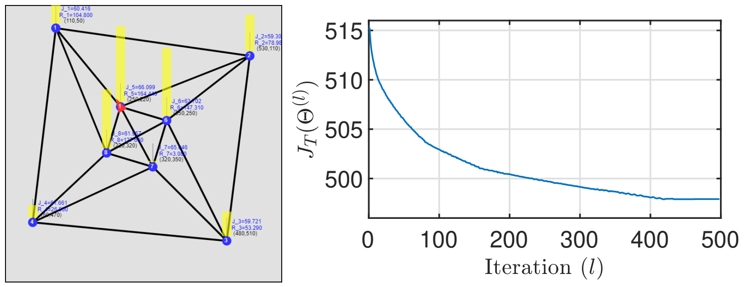
**(a)** Config. at $t = T$. **(b)** Cost vs iterations plot.

**Figure 4·18:** Single-agent simulation example 2 (SASE2): Started with a random $\Theta^{(0)}$ and converged to a TCP with $J_T = 651.3$.



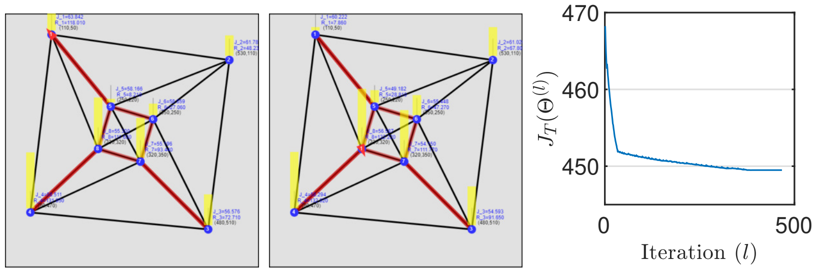
**(a)** $l = 0$, $t = T$. **(b)** $l = 100$, $t = T$. **(c)** Cost vs iterations.

**Figure 4·19:** SASE2: The derived initial TCP $\Theta^{(0)}$ has a cost $J_T = 607.9$ and is further improved by (4.6) to reach a TCP with a cost $J_T = 567.0$ (Improvement $= +84.3$ compared to Fig. 4·18).



**(a)** Config. at $t = T$. **(b)** Cost vs iterations plot.

**Figure 4·20:** Single agent simulation example 3 (SASE3): Started with a random $\Theta^{(0)}$ and converged to a TCP with $J_T = 497.9$.

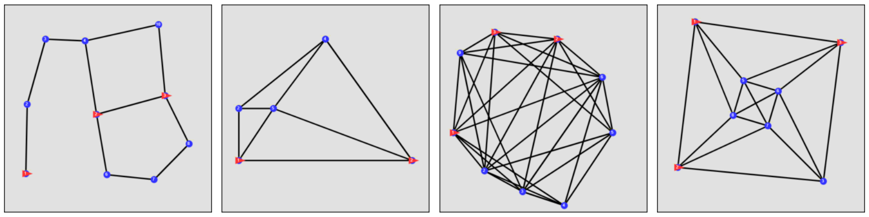


**(a)** $l = 0$, $t = T$. **(b)** $l = 500$, $t = T$ **(c)** Cost vs iterations.

**Figure 4·21:** SASE3: The derived initial TCP $\Theta^{(0)}$ has a cost $J_T = 468.2$ and is further improved by (4.6) to reach a TCP with a cost $J_T = 449.5$ (Improvement $= +48.4$ compared to Fig. 4·20).

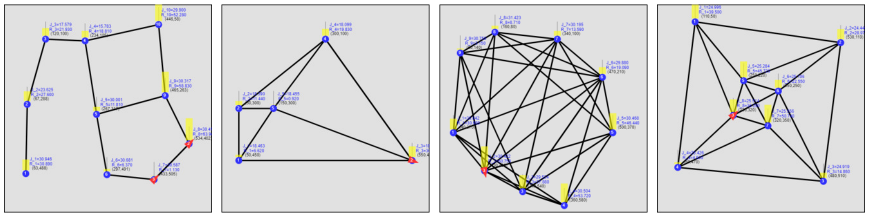
Next, the four multi-agent PMN problem configurations shown in Figs. 4·22(a)-(d) (labeled MASE1-MASE4) are considered. Note that in MASE2, only two agents were deployed, whereas in all the rest three agents were deployed. Figure 4·23 shows the respective $J_T$ values attained from (4.6) when initialized with randomly selected $\Theta^{(0)}$. The sub-graphs obtained from the proposed graph partitioning technique (Step 1 of Alg. 4.1) for each of the MASEs are shown in Fig. 4·24. The constructed target-cycles in sub-graphs and the process of sub-graph refinement (Steps 2 and 3 of Alg. 4.1) are demonstrated in Figs. 4·25(a)-(d) with regard to the MASE1. Sub-figures in Fig. 4·26 show the determined respective graph partitions and target-cycles. It was observed that the initial TCP given by Alg. 4.1 is directly locally optimal in each MASE. However, each of these TCPs performed better than the optimal TCPs obtained with randomly initialized $\Theta^{(0)}$ (shown in Fig. 4·23). In particular, the percentage improvements achieved are: 66.3%, 61.7% 78.2%, and 70.3%, respectively. All the discussed simulation results so far have been summarized in Table 4.1.

Finally, eight randomly generated MASEs are considered (with $N = 3$, $M = 15$, see (Welikala and Cassandras, 2019a) for details). When the proposed PMN solution (i.e., Alg. 4.1) was deployed, across these eight MASEs, the average percentage



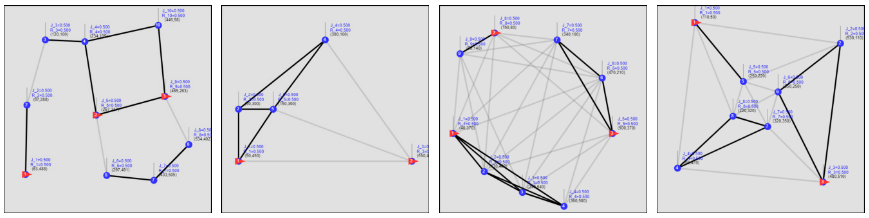
**(a)** MASE1 **(b)** MASE2 **(c)** MASE3 **(d)** MASE4

**Figure 4·22:** Multi-agent simulation examples (MASEs) at $t = 0$.



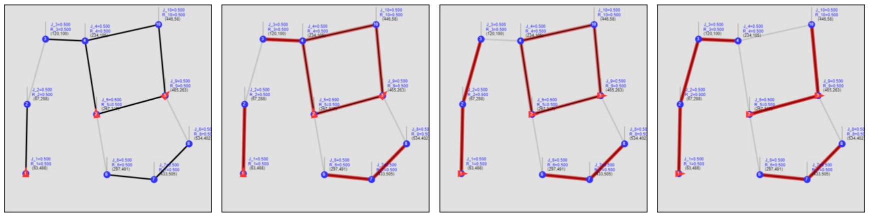
**(a)** MASE1 $J_T = 270.2$ **(b)** MASE2 $J_T = 91.7$ **(c)** MASE3 $J_T = 274.0$ **(d)** MASE4 $J_T = 201.3$

**Figure 4·23:** Cost $J_T$ achieved in each MASE upon convergence when started with a random $\Theta^{(0)}$ (Config. at $t = T$ is shown).



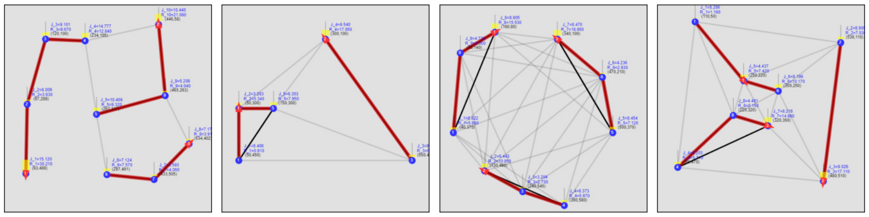
**(a)** MASE1 **(b)** MASE2 **(c)** MASE3 **(d)** MASE4

**Figure 4·24:** Clustering results obtained for the considered MASEs.



**(a)** Clusters **(b)** Cycles **(c)** Trade 1 **(d)** Trade 2

**Figure 4·25:** MASE1: (a) initial sub-graphs, (b) initial target-cycles, (c)-(d) two 'trading' steps and (d) final sub-graphs/target-cycles.



**(a)** MASE1 $J_T = 90.9$ Impr.= +179.3 **(b)** MASE2 $J_T = 35.1$ Impr.= +56.6 **(c)** MASE3 $J_T = 59.5$ Impr.= +214.5 **(d)** MASE4 $J_T = 59.8$ Impr.= +141.5

**Figure 4·26:** Cost $J_T$ and improvement achieved in each MASE compared to Fig. 4·23. Each MASE started (4.6) with the TCP $\Theta^{(0)}$ given by Alg. 4.1 and found that $\Theta^{(0)}$ is directly locally optimal.

**Table 4.1:** A summary of obtained simulation results.

| Cost of the optimal TCP $\Theta^*$ (found using (4.6)): $J_T(\Theta^*)$ | Single-Agent Simulation Examples | | | Multi-Agent Simulation Examples | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 | 4 |
| With randomly generated initial TCP $\Theta^{(0)}$ | 129.2 | 651.3 | 497.9 | 270.2 | 91.7 | 274.0 | 201.3 |
| With initial TCP $\Theta^{(0)}$ given by the proposed Alg. 4.1 | 114.6 | 567.0 | 449.5 | 90.9 | 35.1 | 59.5 | 59.8 |
| Percentage improvement (%) | 11.1 | 12.9 | 9.7 | 66.3 | 61.7 | 78.2 | 70.3 |

improvement achieved was 69.1%. Further, on an Intel® Core™ i7-7800 CPU 3.20 GHz Processor with a 32 GB RAM, the average execution time taken for the proposed Alg. 4.1 to generate the TCP $\Theta^{(0)}$ was 13.7$s$ and all such generated TCPs were immediately locally optimal. In contrast, when the TCPs $\Theta^{(0)}$ were randomly generated, the average execution time observed for the convergence of the gradient descent (4.6) was 245.8$s$. Hence, the execution time taken for the proposed off-line greedy initialization process is much smaller and, at the same time, highly effective.

## 4.6 Summary

This chapter of the thesis addressed the issue of local optima arising in *persistent monitoring on networks* problems (i,e., a class of cooperative multi-agent control problems). First, a class of distributed threshold-based parametric controllers was adopted where IPA can be used to determine the optimal threshold parameters in an on-line manner using gradient descent. Due to the non-convex nature of the PMN problem, the optimal thresholds given by the gradient descent highly depend on the used initial thresholds. To address this issue, the asymptotic behavior of the persistent monitoring system was studied, and based on the findings, a computationally efficient and effective threshold initialization scheme was proposed. Extensive numerical results were provided to show that such systematically chosen initial thresholds while having significantly improved performance levels compared to the state of the art, are almost immediately (locally) optimal or quickly lead to optimal values.

# Chapter 5

# Event-Driven Receding Horizon Control for Persistent Monitoring in Networks

As introduced in Section 1.3.3, this chapter of the thesis proposes an alternative approach to the class of *persistent monitoring on networks* problems discussed in the previous chapter. In particular, as opposed to taking a gradient-based parametric control approach, a gradient-free *event-driven receding horizon control* solution (see Section 1.3.4) is proposed for this class of PMN problems. This RHC approach has many attractive features like being computationally efficient, distributed, on-line, and parameter-free. Numerical results are provided showing improvements compared to the distributed on-line solution proposed in (Zhou et al., 2019).

Sections of this chapter are organized as follows. Section 5.1 presents the problem formulation, preliminary results and an overview of the RHC solution. In Section 5.2, each *receding horizon control problem* form is explicitly solved and in Section 5.3 two possible modifications to the RHCPs are discussed. The performance and robustness of the proposed RHC method are illustrated through simulation results reported in Section 5.4. Finally, Section 5.5 concludes the chapter.

## 5.1 Problem Formulation

Since this chapter considers the same PMN problem as the previous chapter (introduced in its Section 4.1), in the sequel, some intricate details of this PMN problem setup are omitted to avoid unnecessary repetition. However, all the essential information is provided here to make this chapter self-contained.

An $n$-dimensional mission space containing $M$ targets (nodes) in the set $\mathcal{T} = \{1, 2, \dots, M\}$ and $N$ agents in the set $\mathcal{A} = \{1, 2, \dots, N\}$ is considered. The location of target $i \in \mathcal{T}$ is fixed at $Y_i \in \mathbb{R}^n$. Each agent $a \in \mathcal{A}$ is allowed to move within this mission space, and thus its location at time $t$ is denoted by $s_a(t) \in \mathbb{R}^n$.

**Target Model:** The *uncertainty state* $R_i(t) \in \mathbb{R}$ associated with the target $i \in \mathcal{T}$ follows the dynamics (Zhou et al., 2019) (identical to (4.1)):

$$\dot{R}_i(t) = \begin{cases} A_i - B_i N_i(t) & \text{if } R_i(t) > 0 \text{ or } A_i - B_i N_i(t) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

where $A_i, B_i$ and $R_i(0)$ are prespecified and $N_i(t) = \sum_{a \in \mathcal{A}} \mathbf{1}\{s_a(t) = Y_i\}$. Recall that $A_i \in \mathbb{R}_{\geq 0}$, $B_i \in \mathbb{R}_{\geq 0}$ and $N_i(t)$ respectively represents the uncertainty growth rate, the uncertainty removal rate and the number of agents present at target $i$ at time $t$.

**Graph Topology:** A directed graph topology $\mathcal{G} = (\mathcal{T}, \mathcal{E})$ is embedded into the mission space such that the *targets* are represented by the graph *vertices* $\mathcal{T} = \{1, 2, \dots, M\}$ and the inter-target *trajectory segments* are represented by the graph *edges* $\mathcal{E} \subseteq \{(i, j) : i, j \in \mathcal{T}\}$. Recall that each trajectory segment represented by an edge $(i, j) \in \mathcal{E}$ is assigned a (predefined) value $\rho_{ij} \in \mathbb{R}_{>0}$ representing the *travel-time* an agent spends to travel from target $i$ to $j$. Similar to before, based on $\mathcal{E}$, the *neighbor-set* and the *neighborhood* of a target $i \in \mathcal{T}$ are defined respectively as

$$\mathcal{N}_i \triangleq \{j : (i, j) \in \mathcal{E}\} \text{ and } \bar{\mathcal{N}_i} \triangleq \mathcal{N}_i \cup \{i\}. \tag{5.2}$$

Similar to the previous chapter, the analysis presented in this chapter is independent of the agent motion dynamic model which ultimately determines the values of $\rho_{ij}$ for edges $(i, j) \in \mathcal{E}$. However, note that the ongoing research extends this model by considering $\rho_{ij}$ as functions of controllable motion variables (e.g., speed, acceleration) (Welikala and Cassandras, 2021d).

**Objective:** The objective is to minimize the *mean system uncertainty* $J_T$ over a finite time interval $[0, T]$ (identical to (4.2)):

$$J_T \triangleq \frac{1}{T}\int_0^T \sum_{i\in\mathcal{T}} R_i(t)dt, \tag{5.3}$$

by controlling the motion of the team of agents through a suitable set of feasible distributed controllers described next.

**Control:** Note that whenever an agent $a \in \mathcal{A}$ is ready to leave a target $i \in \mathcal{T}$, its *next-visit* target $j$ is selected from $\mathcal{N}_i$. Next, the agent travels on the trajectory segment $(i, j) \in \mathcal{E}$ to arrive at target $j$ spending a travel-time $\rho_{ij}$. Subsequently, it selects a *dwell-time* $\tau_j \in \mathbb{R}_{\geq 0}$ to spend at target $j$ (which contributes to decreasing $R_j(t)$), and then makes another next-visit decision.

Therefore, in a PMN problem, the control exerted on an agent consists of a sequence of *next-visit* targets $j \in \mathcal{N}_i$ and *dwell-times* $\tau_i \in \mathbb{R}_{\geq 0}$. The goal is to determine the decisions $(\tau_i, j)$ for any agent residing at any target $i$ at any time $t \in [0, T]$ which are collectively optimal in the sense of minimizing (5.3). As noted in the previous chapter, this is a challenging task due to the nature of the feasible control space.

**Receding Horizon Control:** As opposed to the proposed parametric controller in the previous chapter, this chapter proposes an *Event-Driven Receding Horizon Controller* (RHC) for each agent $a \in \mathcal{A}$. As introduced in Section 1.3.4, an event-driven receding horizon controller solves an optimization problem of the form (5.3) but limited to a prespecified *planning horizon* whenever an *event* is observed; the resulting (optimal) control is then executed over a generally shorter *action horizon* defined by the occurrence of the next event of interest to the controller. This process is iteratively repeated in an event-driven fashion.

In the PMN problem, the aim of the RHC, when invoked at time $t$ for an agent residing at target $i \in \mathcal{T}$, is to determine the immediate next-visit target $j \in \mathcal{N}_i$ and dwell-times at targets $i$ and $j$ (i.e., $\tau_i$ and $\tau_j$, respectively). These three decisions jointly form a control $U_i(t)$, and its optimal value is determined by solving an optimization problem of the form:

$$U_i^*(t) = \underset{U_i(t) \in \mathbb{U}(t)}{\arg\min} \left[ J_H(X_i(t), U_i(t); H) + \hat{J}_H(X_i(t+H)) \right], \tag{5.4}$$

where $X_i(t)$ is the current local state and $\mathbb{U}(t)$ is the *feasible control set* at $t$ (whose exact definition will be provided later). The term $J_H(X_i(t), U_i(t); H)$ is the immediate cost over the planning horizon $[t, t+H]$ and $\hat{J}_H(X_i(t+H))$ is an estimate of the future cost evaluated at the end of the planning horizon $t + H$.

In prior work (Li and Cassandras, 2006; Khazaeni and Cassandras, 2018b; Chen and Cassandras, 2020b), the value of the planning horizon length $H$ is selected *exogenously*. However, in this work, it is included into the optimization problem and the $\hat{J}_H(X_i(t + H))$ term is ignored. Thus, by optimizing the planning horizon, the proposing RHC approach compensates for the complexity and intrinsic inaccuracy of the $\hat{J}_H(X_i(t+H))$ term whose evaluation requires global information. It is important to point out that the proposed RHC approach is distributed and on-line as it allows each agent to separately solve (5.4) using only the current local state information.

### 5.1.1 Preliminary Results

According to (5.1), the target state $R_i(t)$, $i \in \mathcal{T}$, is piece-wise linear and its gradient $\dot{R}_i(t)$ changes only when one of the following (*strictly local* to target $i$) *events* occurs: (i) An agent arrival at $i$, (ii) An event $[R_i \to 0^+]$, or (iii) An agent departure from $i$.

Let the occurrence of such events associated with target $i$ be indexed by $k = 1, 2, \ldots$ and the respective event occurrence times be denoted by $t_i^k$ with $t_i^0 = 0$. Then,

$$\dot{R}_i(t) = \dot{R}_i(t_i^k), \quad \forall t \in [t_i^k, t_i^{k+1}). \tag{5.5}$$

As pointed out in the previous chapter, allowing overlapping dwell intervals at some target (also referred to as "simultaneous target sharing") is known to lead to solutions with poor performance levels (clearly, this issue only applies if $N > 1$). This observation motivates enforcing a constraint on the controller to ensure:

$$N_i(t) \in \{0, 1\}, \ \forall t \in [0, T], \ \forall i \in \mathcal{T}. \tag{5.6}$$

If the control constraint (5.6) is enforced, it follows from (5.1) and (5.5) that the sequence $\{\dot{R}_i(t_i^k)\}$, $k = 0, 1, 2, \ldots$, is a *cyclic order* of three elements: $\{-(B_i - A_i), 0, A_i\}$. Next, in order to ensure that each agent is capable of enforcing the event $[R_i \to 0^+]$ at any $i \in \mathcal{T}$, the following simple stability condition is assumed.

**Assumption 5.1.** *Target uncertainty rate parameters $A_i$ and $B_i$ of each target $i \in \mathcal{T}$ satisfy $0 \leq A_i < B_i$.*

Note that this condition is less restrictive compared to the condition: $\sum_{i=1}^{m} \frac{A_i}{B_i} \leq 1$ (where $m \geq 2$) assumed in Theorem 4.1 in the previous chapter.

**Decomposition of the Objective Function:** Let the contribution of target $i$ to the objective $J_T$ in (5.3) during a time period $[t_0, t_1)$ be $\frac{1}{T} J_i(t_0, t_1)$ where

$$J_i(t_0, t_1) \triangleq \int_{t_0}^{t_1} R_i(t) dt.$$

Theorem 5.1 provides a target-wise and temporal decomposition of the main objective function $J_T$ in (5.3) based on $J_i(t_0, t_1)$.

**Theorem 5.1.** *The contribution to the main objective $J_T$ by target $i \in \mathcal{T}$ during a time period $[t_0, t_1) \subseteq [t_i^k, t_i^{k+1})$ for some $k \in \mathbb{Z}_{\geq 0}$ is $\frac{1}{T} J_i(t_0, t_1)$, where,*

$$J_i(t_0, t_1) = \frac{(t_1 - t_0)}{2} \Big[ 2R_i(t_0) + \dot{R}_i(t_0)(t_1 - t_0) \Big]. \tag{5.7}$$

*Proof.* See Appendix A.5.1. □

A simple corollary of Theorem 5.1 is to extend it to any interval $[t_0, t_1)$ which may include one or more event times $t_i^k$.

**Corollary 5.1.** *Let $t_0 = t_i^k$ be the time when an agent arrived at target $i \in \mathcal{T}$, followed by an $[R_i \to 0^+]$ event at $t = t_i^{k+1}$ and a departure event at $t = t_i^{k+2}$. Then, for any $t_1$ such that $t_i^{k+2} \leq t_1 \leq t_i^{k+3}$,*

$$J_i(t_0, t_1) = \frac{u_i^0}{2} \left[ 2R_i(t_0) - (B_i - A_i)u_i^0 \right] + \frac{u_i^1}{2} \left[ A_i u_i^1 \right], \tag{5.8}$$

*where $u_i^0 = t_i^{k+1} - t_0$ and $u_i^1 = t_1 - t_i^{k+2}$.*

*Proof:* The result immediately follows by applying Theorem 5.1 to the three inter-event intervals $[t_0, t_i^{k+1})$, $[t_i^{k+1}, t_i^{k+2})$ and $[t_i^{k+2}, t_1)$ and then using the $\dot{R}_i$ values stated earlier and the fact that $R_i(t_i^{k+1}) = R_i(t_i^{k+2}) = 0$. ■

**Local Objective Function:** The *local objective function* of target $i$ over a time period $[t_0, t_1) \subseteq [0, T]$ is defined as

$$\bar{J}_i(t_0, t_1) \triangleq \sum_{j \in \bar{\mathcal{N}}_i} J_j(t_0, t_1). \tag{5.9}$$

The value of each $J_j(t_0, t_1)$ term above is obtained through Theorem 5.1 and its extension in Corollary 5.1 if $[t_0, t_1)$ includes additional events (where $[t_0, t_1)$ is decomposed into a sequence of corresponding inter-event time intervals).

In developing a distributed event-driven controller for an agent residing at some target $i \in \mathcal{T}$, it is reasonable to assume that this agent has access to any necessary *local* information from the neighborhood $\bar{\mathcal{N}}_i$. Therefore, $\bar{J}_i(t_0, t_1)$ can be evaluated by this agent at any required (event-driven) time instant.

### 5.1.2 RHC Problem (RHCP) Formulation

Consider a situation where agent $a \in \mathcal{A}$ resides at target $i \in \mathcal{T}$ at some $t \in [0, T]$. In the considered distributed setting, it is assumed that agent $a$ is made aware of only *local events* occurring in the neighborhood $\bar{\mathcal{N}}_i$. As defined earlier, the control $U_i(t)$ consists of (i) the *dwell-time* $\tau_i$ at the current target $i$, (ii) the *next-visit* target $j \in \mathcal{N}_i$ and (iii) the *dwell-time* $\tau_j$ at the selected next-visit target $j$. Moreover, note that a dwell-time decision $\tau_i$ (or $\tau_j$) can be divided into two interdependent decisions: (i) the *active time* $u_i$ (or $u_j$) when $R_i(t) > 0$ ($R_j(t) > 0$) and (ii) the *inactive* (or *idle*) *time* $v_i$ (or $v_j$) when $R_i(t) = 0$ ($R_j(t) = 0$), as shown in Fig. 5·1. Thus, agent $a$ has to optimally choose five decision variables which form the control vector $U_i(t) \triangleq [u_i(t), v_i(t), j(t), u_j(t), v_j(t)]$. Note that $j(t)$ is discrete while the remaining four components of $U_i(t)$ are real-valued. The time argument of each component of $U_i(t)$ is omitted henceforth for notational convenience.

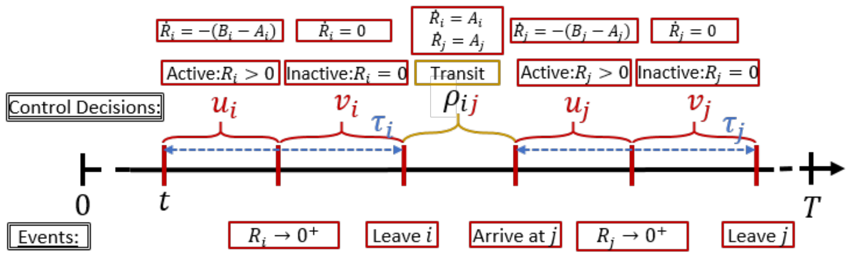


**Figure 5·1:** Event timeline and control decisions under RHC.

**Fixed Planning Horizon:** Recalling (5.4), the RHC depends on the planning horizon $H \in \mathbb{R}_{\geq 0}$ which is normally viewed as a *fixed* control parameter. Intuitively, selecting $H \geq \max_{(i,j)\in\mathcal{E}} \rho_{ij}$ ensures that all agents will consider traveling to all of their current neighboring targets. However, note that $t+H$ is constrained by $t+H \leq T$; hence, if this is violated, the planning horizon should be truncated to be $H = T - t$.

In (5.4), let the current local state be $X_i(t) = \{R_m(t) : m \in \bar{\mathcal{N}_i}\}$ and decompose the control $U_i(t)$ into its real-valued components and its discrete component (omitting time arguments) as $U_{ij} \triangleq [u_i, v_i, u_j, v_j] \in \mathbb{U}$ and $j \in \mathcal{N}_i$, respectively. Now, if the objective function $J_H(\cdot)$ in (5.4) is chosen to reflect the contribution to the main objective $J_T$ in (5.3) by the targets in the neighborhood $\bar{\mathcal{N}_i}$ over the fixed time period $[t, t+H]$ (which is provided by (5.9) and Theorem 5.1), then,

$$\begin{aligned} &J_H(X_i(t), U_{ij}; H) = \frac{1}{H}\ \bar{J}_i(t, t+H) \text{ with} \\ &\mathbb{U} = \{U : U \in \mathbb{R}^4,\ U \geq 0,\ |U| + \rho_{ij} = H\}. \end{aligned} \tag{5.10}$$

The feasible control set $\mathbb{U}$ (of (5.4)) is such that $u_i$, $v_i$, $u_j$, and $v_j$ are non-negative real variables. Note that the notation $|\cdot|$ is used to represent the 1-norm or the cardinality operator when the argument is respectively a vector or a set.

In this setting, the optimal controls are obtained by solving the following set of optimization problems, henceforth called the RHC Problem (RHCP):

$$U_{ij}^* = \underset{U_{ij} \in \mathbb{U}}{\arg\min}\ J_H(X_i(t), U_{ij}; H);\ \forall j \in \mathcal{N}_i \quad \text{and} \tag{5.11}$$

$$j^* = \underset{j \in \mathcal{N}_i}{\arg\min}\ J_H(X_i(t), U_{ij}^*; H). \tag{5.12}$$

Observe that (5.11) involves solving $|\mathcal{N}_i|$ optimization problems, one for each $j \in \mathcal{N}_i$. Then, (5.12) determines $j^*$ through a simple numerical comparison (similar to a greedy step). Therefore, the final optimal decision variables are $U_{ij^*}^*$ and $j^*$.

According to (5.10), the choices for the four control variables in $U_{ij}$ are restricted by $U_{ij} \in \mathbb{U}$ such that $|U_{ij}| + \rho_{ij} = H$ (see also Fig.5·1). Therefore, the selection of $H$ directly affects the RHCP's optimal solution. For example, if $H$ is very large (or very small), clearly the optimal decisions $U_{ij^*}^*$ and $j^*$ are not globally optimal. Attempting to find the optimal choice of $H$ without compromising the on-line distributed nature of the proposed RHC solution is a challenging task.

**Variable Planning Horizon:** This problem is addressed by introducing a *variable horizon* $w$ defined as:

$$w \triangleq |U_{ij}| + \rho_{ij} = u_i + v_i + \rho_{ij} + u_j + v_j, \tag{5.13}$$

and replacing $H$ in (5.10) by $w$ while, at the same time, imposing the constraint $w \leq H$. It is important to observe that $w$ defined in (5.13) is a function of $u_i(t)$,$v_i(t)$,$u_j(t)$ and $v_j(t)$. However, for notational convenience, this dependence is not shown explicitly. It is also important to note that now the value of $H$ is not critical as long as it is sufficiently large; for instance, it can be chosen to be $T-t$. Therefore, the solution of the RHCP (5.11)-(5.12) can now be obtained without any tunable parameters, making the resulting controller *parameter-free.* The objective function $J_H$ and the feasible control set $\mathbb{U}$ in the RHCP are now chosen as

$$\begin{aligned} &J_H(X_i(t), U_{ij}; H) = \frac{1}{w}\bar{J}_i(t, t+w) \text{ and} \\ &\mathbb{U} = \{U : U \in \mathbb{R}^4,\ U \geq 0,\ |U| + \rho_{ij} \leq H\}. \end{aligned} \tag{5.14}$$

Therefore, this novel RHCP formulation allows the simultaneously determination of the *optimal planning horizon* size $w^*$ in terms of the optimal control $U_i^*(t)$ as

$$w^* = |U_{ij^*}^*| + \rho_{ij^*}. \tag{5.15}$$

On the other hand, this incorporation of $w$ in (5.14), as opposed to (5.10), makes the denominator term of the objective function control-dependent and introduces new technical challenges that will be addressed in the rest of this chapter. To accomplish this, structural properties of (5.14) are exploited and it is shown that the RHCP in

(5.11) can be solved analytically and efficiently to obtain its globally optimal solution.

**Event-Driven Action Horizon:** As in all receding horizon controllers, the solution of each optimization problem over a certain planning horizon is executed only over a shorter *action horizon* $h$. In the distributed RHC setting, the value of $h$ is determined by the first event that the agent observes after $t$, the time instant when the agent last solved the RHCP. Thus, in contrast to time-driven receding horizon control, the RHC solution is updated whenever asynchronous events occur; this prevents unnecessary steps to re-solve the RHCP (5.11)-(5.12) with (5.14).

Figure 5·2 shows an example of three consecutive action horizons (labeled $h_1, h_2$ and $h_3$) observed by an agent $a$ after an event at $t$ triggers the solution of the RHCP. Note that $w_1^*$, $w_2^*$, $w_3^*$ represent the three optimal planning horizon sizes (i.e., $w^*$ in (5.15)) determined at each respective local event time $t$, $t + h_1$ and $t + h_1 + h_2$.

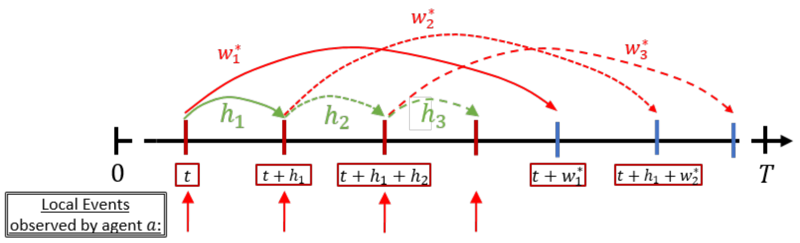


**Figure 5·2:** Event driven receding horizon control approach.

In general, the determination of the action horizon $h$ may be controllable or uncontrolled. The latter case occurs as a result of random events in the system (if such events are part of the setting), while the former corresponds to the occurrence of any one event whose occurrence results from an agent solving a RHCP at some earlier time. Next, the three *controllable* events associated with an agent when it resides at target $i$ are defined. Each of these events defines the action horizon $h$ following the solution of a RHCP by this agent at some time $t \in [0, T]$:

**1. Event** $[h \to u_i^*]$**:** This event occurs at time $t + u_i^*(t)$. If $R_i(t + u_i^*(t)) = 0$, this event coincides with an $[R_i \to 0^+]$ event. Otherwise, $R_i(t + u_i^*(t)) > 0$ implies that the solution of the associated RHCP dictates ending the active time at target $i$ before the $[R_i \to 0^+]$ event. Hence, in that case, by definition, no inactive time may follow, i.e., $v_i^*(t) = 0$, and $[h \to u_i^*]$ coincides with a departure event from target $i$.

**2. Event** $[h \to v_i^*]$**:** This event occurs at time $t + v_i^*(t)$. It is only feasible after an event $[h \to u_i^*]$ has occurred, including the possibility that $u_i^*(t) = 0$ in the RHCP solution determined at $t$. Clearly, this event always coincides with an agent departure event from $i$.

**3. Event** $[h \to \rho_{ij^*}]$**:** This event occurs at time $t + \rho_{ij^*(t)}$. It is only feasible after an event $[h \to u_i^*]$ or $[h \to v_i^*]$ has occurred, including the possibility that $u_i^*(t) = 0$ and $v_i^*(t) = 0$ in the RHCP solution determined at $t$. Clearly, this coincides with an arrival event at target $j^*(t)$ as determined by the RHCP solution obtained at time $t$.

Observe that these events are mutually exclusive, i.e., only one is feasible at any one time. In addition, there are uncontrollable events associated with neighboring targets in $\mathcal{N}_i$ other than target $i$. In particular, two additional events are defined that may occur at any neighbor $j \in \mathcal{N}_i$ and trigger an event at the agent residing at target $i$. These events have been designed to enforce the no-simultaneous-target-sharing policy (the control constraint (5.6)) and apply only to multi-agent PMN problems.

A target $j \in \mathcal{T}$ is said to be *covered* at time $t$ if it already has a residing agent or if an agent is en route to visit it from a neighboring target in $\mathcal{N}_j$. Since neighboring targets communicate with each other, this information can be determined at any target in $\bar{\mathcal{N}}_j$ at any time $t$. Therefore, an agent $a \in \mathcal{A}$ residing at target $i$ can prevent target sharing at target $j \in \mathcal{N}_i$ by simply modifying the neighbor set $\mathcal{N}_i$ used in the RHCP solved at time $t$ to exclude all covered targets. Here, $\mathcal{N}_i(t)$ is used to indicate a *time-varying* neighborhood of target $i$. Then, if target $j$ becomes covered at time $t$,

$$\mathcal{N}_i(t) = \mathcal{N}_i(t^-)\backslash\{j\}, \tag{5.16}$$

is set. The effect of this modification is clear if a RHCP solved by an agent at target $i$ at some time $t$ leads to a next-visit solution $j^* \in \mathcal{N}_i(t)$ and if this is followed by an event at $t' > t$ causing target $j^*$ to become covered, then $\mathcal{N}_i(t') = \mathcal{N}_i(t)\backslash\{j^*\}$ and the agent at target $i$ (whether active or inactive) must re-solve the RHCP at $t'$ with the new $\mathcal{N}_i(t')$. Note that as soon as an agent $a$ is en route to $j^*$, then $j^*$ becomes covered, hence preventing any other agent from visiting $j^*$ prior to agent $a$'s subsequent departure from $j^*$.

Based on this discussion, the following two additional *neighbor-induced local events* are defined that are triggered at $j \in \mathcal{N}_i$ and affecting an agent $a$ residing at target $i$:

**4. Covering Event $C_j$, $j \in \mathcal{N}_i$:** This event causes $\mathcal{N}_i(t)$ to be modified to $\mathcal{N}_i(t^-)\backslash\{j\}$.

**5. Uncovering Event $\bar{C}_j$, $j \in \mathcal{N}_i$:** This event causes $\mathcal{N}_i(t)$ to be modified to $\mathcal{N}_i(t^-) \cup \{j\}$.

If one of these two events takes place while an agent residing at target $i$ is either active or inactive, then the RHCP (5.11)-(5.12) is re-solved (replacing $\mathcal{N}_i$ with $\mathcal{N}_i(t)$) to account for the updated $\mathcal{N}_i(t)$.

**Three Forms of the RHCP:** It is clear from this discussion that the exact form of the RHCP to be solved at time $t$ depends on the event that triggered the end of the previous action horizon (i.e., the event occurring at time $t$) and the target state $R_i(t)$. In particular, there are three possible forms of the RHCP (5.11)-(5.12):

**- RHCP1:** This problem is solved by an agent arriving at target $i$, i.e., when an event $[h \to \rho_{ki}]$ occurs at time $t$ for any $k \in \mathcal{N}_i(t)$. The solution $U_i^*(t)$ includes $u_i^*(t) \geq 0$, representing the amount of time that the agent should be active at target $i$. This problem is also solved while the agent is active at target $i$ (i.e., while $R_i(t) > 0$) if a $C_j$ or $\bar{C}_j$ event occurs for any $j \in \mathcal{N}_i(t)$.

**- RHCP2:** This problem is solved by an agent residing at target $i$ when an event $[h \to u_i^*]$ occurs at time $t$ with $R_i(t) = 0$. This problem is also solved while the agent is inactive (i.e., $R_i(t) = 0$) at $i$ if a $C_j$ or $\bar{C}_j$ event occurs for any $j \in \mathcal{N}_i(t)$. In both cases, the solution $U_i^*(t)$ is now constrained to include $u_i^*(t) = 0$ by default, since the agent can no longer be active at target $i$.

**- RHCP3:** This problem is solved by an agent departing from target $i$ and may be triggered by one of two events: (i) Event $[h \to u_i^*]$ at time $t$ with $R_i(t) > 0$. The solution $U_i^*(t)$ is constrained to include $u_i^*(t) = 0$ by default; in addition, it is constrained to have $v_i^*(t) = 0$ since the agent ceases being active while $R_i(t) > 0$, implying that it must immediately depart from target $i$ without becoming inactive. (ii) Event $[h \to v_i^*]$ at time $t$, implying that the agent is no longer inactive and must depart from target $i$. As in case (i), the solution $U_i^*(t)$ is constrained to have both $u_i^*(t) = 0$ and $v_i^*(t) = 0$ by default.

**Complexity of RHCPs:** As will be shown next, all three problem forms of the RHCP discussed above can be solved to obtain the corresponding globally optimal solutions in closed form. Therefore, their complexity is constant and the overall RHC complexity scales linearly with the number of events occurring in $[0, T]$.

## 5.2 Solving Event-Driven RHCPs

This section presents the solutions to the identified three forms of RHCPs discussed above. Due to its relative simplicity, **RHCP3** is considered first.

### 5.2.1 Solution of RHCP3

Recall that an agent solves **RHCP3** when it is ready to leave the target where it resides. Therefore, $u_i^*(t) = 0$ and $v_i^*(t) = 0$ by default and $U_{ij}$ in (5.11) is reduced to $U_{ij} = [u_j, v_j]$ with $U_i(t) = [j, u_j, v_j]$. The obtained $j^*(t)$ directly defines the next destination to visit. Clearly, **RHCP3** plays a crucial role in defining agent trajectories in terms of targets visited. The variable horizon $w$ (5.13) for this case is $w = \rho_{ij} + u_j + v_j$ and, from (5.14), $w$ is constrained so that $\rho_{ij} \leq w \leq H$. Therefore, $\rho_{ij} \leq H$ is assumed henceforth in this section.

**Constraints:** First, an upper bound for the active time control variable $u_j$ is identified. This is defined by the the maximum active time possible at target $j$, which is given by $R_j(t + \rho_{ij} + u_j) = 0$. Denoting this bound by $u_j^B$, it follows from (5.1) that

$$u_j^B(t) \triangleq \frac{R_j(t + \rho_{ij})}{B_j - A_j} = \frac{R_j(t) + A_j \rho_{ij}}{B_j - A_j}. \tag{5.17}$$

Note that the dependence of $u_j^B(t)$ on $t$ captures its dependence on the initial condition $R_j(t)$; for notational simplicity, this time dependence is henceforth omitted. A tighter upper-bound than $u_j^B$ on $u_j$, as well as an upper-bound on $v_j$, denoted respectively by $\bar{u}_j$ and $\bar{v}_j$ are imposed by the variable horizon constraint $w = u_j + v_j + \rho_{ij} \leq H$:

$$\bar{u}_j \triangleq \min\{u_j^B,\ H - \rho_{ij}\} \quad \text{and} \quad \bar{v}_j \triangleq H - (\rho_{ij} + u_j^B). \tag{5.18}$$

Moreover, in order to have a positive inactive time $v_j > 0$ a necessary condition is that it first spends the maximum active time possible $u_j = u_j^B$. Hence, it is clear that any feasible pair $U_{ij} = [u_j, v_j] \in \mathbb{U}$ in (5.11) belongs to one of the two constraint sets:

$$\mathbb{U}_1 = \{0 \leq u_j \leq \bar{u}_j,\ v_j = 0\} \quad \text{or} \quad \mathbb{U}_2 = \{u_j = u_j^B,\ 0 \leq v_j \leq \bar{v}_j\}, \tag{5.19}$$

where $u_j^B$ and $\bar{u}_j, \bar{v}_j$ are given in (5.17) and (5.18), respectively.

**Objective:** Following from (5.14), the objective function corresponding to **RHCP3** is taken as $J_H(U_{ij}) = J_H(X_i(t), [0, 0, U_{ij}]; H) = \frac{1}{w}\bar{J}_i(t, t+w)$. To obtain an exact expression for $J_H(U_{ij})$, first the local objective function $\bar{J}_i$ is decomposed using (5.9):

$$\bar{J}_i = J_j + \sum_{m \in \bar{\mathcal{N}}_i \setminus \{j\}} J_m. \tag{5.20}$$

Considering the state trajectories shown in Fig. 5·3 for the case where agent $a$ goes from target $i$ to target $j$ with decisions $u_j$ and $v_j$, both $J_j$ and $J_m$ terms in (5.20) are evaluated for the period $[t, t+w)$ using Theorem 5.1 as

$$\begin{aligned} J_j &= \frac{\rho_{ij}}{T}[2R_j(t) + A_j\rho_{ij}] + \frac{u_j}{T}[2(R_j(t) + A_j\rho_{ij}) - (B_j - A_j)u_j], \\ J_m &= \frac{(\rho_{ij} + u_j + v_j)}{T}[2R_m(t) + A_m(\rho_{ij} + u_j + v_j)]. \end{aligned}$$

Combining these two results and substituting them in (5.20) gives

$$J_H(u_j, v_j) = \frac{C_1 u_j^2 + C_2 v_j^2 + C_3 u_j v_j + C_4 u_j + C_5 v_j + C_6}{\rho_{ij} + u_j + v_j}, \tag{5.21}$$

where $C_1 = \frac{1}{2}\left[\bar{A} - B_j\right]$, $C_2 = \frac{\bar{A}_j}{2}$, $C_3 = \bar{A}_j$, $C_4 = \left[\bar{R}(t) + \bar{A}\rho_{ij}\right]$, $C_5 = \left[\bar{R}_j(t) + \bar{A}_j\rho_{ij}\right]$, $C_6 = \frac{\rho_{ij}}{2}\left[2\bar{R}(t) + \bar{A}\rho_{ij}\right]$ and

$$\begin{aligned} \bar{A}_{ij} &= \sum_{m \in \mathcal{N}_i \setminus \{j\}} A_m, \quad \bar{A}_i = \bar{A}_{ij} + A_j, \quad \bar{A}_j = \bar{A}_{ij} + A_i, \quad \bar{A} = \bar{A}_{ij} + A_i + A_j, \\ \bar{R}_{ij}(t) &= \sum_{m \in \mathcal{N}_i \setminus \{j\}} R_m(t), \quad \bar{R}_i = \bar{R}_{ij} + R_j, \quad \bar{R}_j = \bar{R}_{ij} + R_i, \quad \bar{R} = \bar{R}_{ij} + R_i + R_j. \end{aligned} \tag{5.22}$$

Note that $C_i \geq 0$ for all $i$ except $C_1$ which is non-negative only when $B_j \leq \bar{A}$.

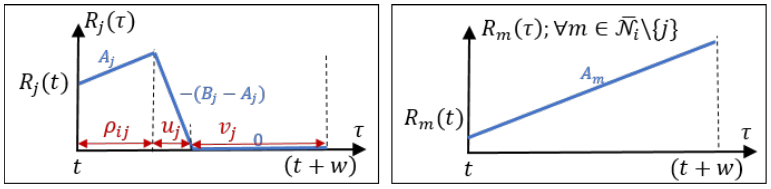


**Figure 5·3:** State trajectories during $[t, t+w)$ for the **RHCP3**.

**Solving RHCP3 for Optimal Control** $(u_j^*, v_j^*)$**:** Based on the first step of RHCP (5.11), $(u_j^*, v_j^*)$ is given by

$$(u_j^*, v_j^*) = \underset{(u_j,v_j)}{\arg\min}\ J_H(u_j, v_j), \tag{5.23}$$

where $(u_j, v_j) \in \mathbb{U}_1$ or $(u_j, v_j) \in \mathbb{U}_2$ as in (5.19).

**- Case 1:** $(u_j, v_j) \in \mathbb{U}_1 = \{0 \leq u_j \leq \bar{u}_j,\ v_j = 0\}$. Clearly, $v_j^* = 0$ and (5.23) takes the form:

$$u_j^* = \underset{0 \leq u_j \leq \bar{u}_j}{\arg\min}\ J_H(u_j, 0). \tag{5.24}$$

**Lemma 5.1.** *The unique optimal solution of* (5.24) *is*

$$u_j^* = \begin{cases} \bar{u}_j & \textit{if } \bar{u}_j \geq u_j^{\#} \textit{ and } \bar{A} < B_j \\ 0 & \textit{otherwise,} \end{cases} \tag{5.25}$$

*where*

$$u_j^{\#} = \frac{\bar{A}\rho_{ij}}{B_i - \bar{A}}. \tag{5.26}$$

*Proof.* See Appendix A.5.2. □

Note that $u_j^{\#}$ in (5.26) is known to the agent and it can be thought of as a break-even point for $u_j$, where if $\bar{u}_j$ allows $u_j$ to increase beyond the $u_j^{\#}$ value, it is always optimal to do so by choosing the extreme point $u_j^* = \bar{u}_j$.

**Remark 5.1.** *When $H$ is sufficiently large, according to* (5.19) *and* (5.17)*,* $\bar{u}_j = u_j^B = \frac{R_j(t)+A_j\rho_{ij}}{B_j-A_j}$*. Therefore, the condition* $\bar{u}_j \geq u_j^{\#}$ *used in* (5.24) *becomes explicitly dependent on the target state* $R_j(t)$*:*

$$u_j^* = \begin{cases} \bar{u}_j & \textit{if } R_j(t) \geq \rho_{ij}\left[\frac{B_j-A_j}{B_j-\bar{A}} \cdot \bar{A} - A_j\right] \textit{ and } \bar{A} < B_j \\ 0 & \textit{otherwise.} \end{cases} \tag{5.27}$$

**- Case 2:** $(u_j, v_j) \in \mathbb{U}_2 = \{u_j = u_j^B,\ 0 \leq v_j \leq \bar{v}_j\}$. Obviously, $u_j^* = u_j^B$ and (5.23) takes the form:

$$v_j^* = \underset{0 \leq v_j \leq \bar{v}_j}{\arg\min}\ J_H(u_j^B, v_j). \tag{5.28}$$

**Lemma 5.2.** *The unique optimal solution of* (5.28) *is*

$$v_j^* = \begin{cases} 0 & \textit{if } \bar{A} \geq B_j\left[1 - \frac{\rho_{ij}^2}{(\rho_{ij}+u_j^B)^2}\right] \\ \min\{v_j^{\#},\ \bar{v}_j\} & \textit{otherwise,} \end{cases} \tag{5.29}$$

*where*

$$v_j^{\#} = \sqrt{\frac{(B_j - A_j)(\rho_{ij} + u_j^B)^2 - B_j\rho_{ij}^2}{\bar{A}_j}} - (\rho_{ij} + u_j^B). \tag{5.30}$$

*Proof.* See Appendix A.5.3. □

Similar to $u_j^{\#}$ in (5.26), $v_j^{\#}$ in (5.30) is completely known to the agent. However, unlike $u_j^{\#}$, $v_j^{\#}$ represents an optimal choice available for $v_j$. Therefore, whenever the constraints on $v_j$ in (5.28) (i.e., $0 \leq v_j \leq \bar{v}_j$) allow it, $v_j^* = v_j^{\#}$ should be chosen.

**Remark 5.2.** *The terms* $u_j^B$ *and* $\bar{v}_j$ *involved in* (5.29) *can be simplified (using* (5.17) *and* (5.19) *respectively) to illustrate the state dependent nature of* $v_j^*$ *as follows:*

$$v_j^* = \begin{cases} 0 & \textit{if } \bar{A} \geq B_j \quad \textit{or } R_j(t) \leq \left[\frac{\rho_{ij}(B_j-A_j)\sqrt{B_j}}{\sqrt{B_j-\bar{A}}} - \rho_{ij}B_j\right] \\ v_j^{\#} & \textit{else if } R_j(t) \leq \left[\sqrt{(B_j-A_j)(H^2\bar{A}_j + \rho_{ij}^2 B_j)} - \rho_{ij}B_j\right] \\ \bar{v}_j & \textit{otherwise.} \end{cases} \tag{5.31}$$

**Theorem 5.2.** *The optimal solution of* (5.23) *is*

$$(u_j^*, v_j^*) = \begin{cases} (0,0) & \textit{if } u_j^\# > \bar{u}_j \textit{ or } \bar{A} \geq B_j \\ (\bar{u}_j, 0) & \textit{else if } \bar{u}_j < u_j^B \\ (u_j^B, 0) & \textit{else if } B_j > \bar{A} \geq B_j \left[1 - \frac{\rho_{ij}^2}{(\rho_{ij}+u_j^B)^2}\right] \\ (u_j^B, v_j^\#) & \textit{else if } v_j^\# \leq \bar{v}_j \\ (u_j^B, \bar{v}_j) & \textit{otherwise,} \end{cases} \tag{5.32}$$

*where* $u_j^\#$ *is given in* (5.26) *and* $v_j^\#$ *is given in* (5.30).

*Proof.* Follows by combining Lemmas 5.1 and 5.2. □

The above theorem implies that whenever: (i) $H$ is sufficiently large (ensuring $\bar{u}_j = u_j^B$ in (5.18)), (ii) the sensing capabilities are high enough to ensure $\bar{A} < B_j$ and (iii) target uncertainty $R_j(t)$ exceeds a known threshold (ensuring $u_j^\# < \bar{u}_j = u_j^B$), it is optimal to select $u_j^* = u_j^B$, hence planning ahead to drive $R_j(t)$ to zero. This conclusion is in line with Theorem 1 in (Zhou et al., 2019).

**Solving for Optimal Next Destination $j^*$:** Using Theorem 5.2, when agent $a$ is ready to leave target $i$ at some local event time $t$, it can compute the optimal trajectory costs $J_H(u_j^*, v_j^*)$ for all $j \in \mathcal{N}_i$. Based on the second step of the RHCP (5.12), the optimal neighbor to visit next is $j^*$ where $j^* = \underset{j \in \mathcal{N}_i}{\arg\min}\ J_H(u_j^*, v_j^*)$.

Thus, upon solving **RHCP3** agent $a$ departs from target $i$ at time $t$ and follows the path $(i, j^*) \in \mathcal{E}$ to visit target $j^*$. This optimal control will be updated upon the occurrence of the next event, which, in this case, will be the arrival event of the agent at $j^*$, triggering the solution of an instance of **RHCP1** at $j^*$.

### 5.2.2 Solution of RHCP2

An agent $a$ residing in target $i$ has to solve **RHCP2** only when an observed event is: (i) $[R_i \to 0^+]$ or (ii) a neighbor induced event $C_j$ or $\bar{C}_j$, $j \in \mathcal{N}_i$, while $R_i(t) = 0$.

Hence, $u_i^*(t) = 0$ by default and $U_i(t) = [v_i, j, u_j, v_j]$ with $U_{ij} = [v_i, u_j, v_j]$ in (5.11). The resulting $v_i^*$ defines the remaining inactive time at target $i$ until the next local event. The variable horizon $w$ in (5.13) for this case is $w = v_i + \rho_{ij} + u_j + v_j$.

**Constraints:** Following the previously used notation, the maximal possible active time at target $j$ forms an upper-bound to the control variable $u_j$ given by

$$u_j^B(v_i) = \frac{R_j(t + v_i + \rho_{ij})}{B_j - A_j} = \frac{R_j(t) + A_j\rho_{ij}}{B_j - A_j} + \frac{A_j}{B_j - A_j} \cdot v_i. \tag{5.33}$$

Note that due to the inclusion of $v_i$ in **RHCP2** (compared to **RHCP3**), $u_j^B$ is now dependent on $v_i$ (see (5.17)). Based on the same arguments as in the analysis of **RHCP3**: (i) to spend a positive inactive time at target $j$, the agent has to first spend the maximum active time possible $u_j^B(v_i)$, and (ii) the variable horizon is subject to $w \leq H$, it is clear that any feasible $U_{ij} = [v_i, u_j, v_j] \in \mathbb{U}$ in (5.11) for **RHCP2** should belong to one of the two constraint sets:

$$\begin{aligned} \mathbb{U}_1 &= \{0 \leq v_i \leq \bar{v}_i(u_j, v_j),\ 0 \leq u_j \leq \bar{u}_j(v_i),\ v_j = 0\}, \\ \mathbb{U}_2 &= \{0 \leq v_i \leq \bar{v}_i(u_j, v_j),\ u_j = u_j^B(v_i),\ 0 \leq v_j \leq \bar{v}_j(v_i)\}, \end{aligned} \tag{5.34}$$

where, $u_j^B(v_i)$ is given in (5.33) and

$$\bar{v}_i(u_j, v_j) = H - (\rho_{ij} + u_j + v_j),$$
$$\bar{u}_j(v_i) = \min\{u_j^B(v_i),\ H - (v_i + \rho_{ij})\} \ \text{ and } \ \bar{v}_j(v_i) = H - (v_i + \rho_{ij} + u_j^B(v_i)).$$

Similar to (5.18), $\bar{u}_j$ and $\bar{v}_j$ respectively represent the limiting values of active and inactive times at $j$. Also, $\bar{v}_i$ is the upper bound to the inactive time at $i$. However, in contrast to (5.18), these three quantities are control-dependent in (5.34). Note also that in (5.34), under $\mathbb{U}_1$, $\bar{v}_i(u_j, v_j) = \bar{v}_i(u_j, 0)$ and under $\mathbb{U}_2$, $\bar{v}_i(u_j, v_j) = \bar{v}_i(u_j^B(v_i), v_j)$.

**Objective:** Following from (5.14), the objective function corresponding to **RHCP2** is $J_H(U_{ij}) = J_H(X_i(t),[0,U_{ij}];H) = \frac{1}{w}\bar{J}_i(t,t+w)$. To obtain an explicit expression for $J_H(U_{ij})$, $\bar{J}_i$ in (5.9) is decomposed as,

$$\bar{J}_i = J_i + J_j + \sum_{m\in\mathcal{N}_i\backslash\{j\}} J_m. \tag{5.35}$$

Considering the trajectories shown in Fig. 5·4 for this case, the three terms $J_i$, $J_j$ and $J_m$ in (5.35) are evaluated for the period $[t,t+w)$ using Theorem 5.1 as

$$\begin{aligned}
J_i =& \frac{A_i(\rho_{ij}+u_j+v_j)^2}{2}\\
J_j =& \frac{(v_i+\rho_{ij})}{2}[2R_j(t)+A_j(v_i+\rho_{ij})] + \frac{u_j}{2}[2(R_j(t)+A_j(v_i+\rho_{ij})) - (B_j-A_j)u_j]\\
J_m =& \frac{(v_i+\rho_{ij}+u_j+v_j)}{2}\,[2R_m(t)+A_m(v_i+\rho_{ij}+u_j+v_j)]
\end{aligned}$$

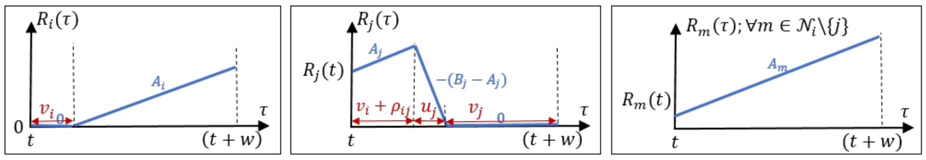


**Figure 5·4:** State trajectories of targets in $\bar{\mathcal{N}}_i$ during $[t,t+w)$.

Combining these results and substituting them in (5.35) gives the complete objective function $J_H(U_{ij})$ as

$$J_H(v_i,u_j,v_j) = \frac{\begin{bmatrix} C_1v_i^2 + C_2u_j^2 + C_3v_j^2 + C_4v_iu_j + C_5v_iv_j \\ + C_6u_jv_j + C_7v_i + C_8u_j + C_9v_j + C_{10}\end{bmatrix}}{v_i+\rho_{ij}+u_j+v_j}, \tag{5.36}$$

where (see also (5.22)) $C_1 = \frac{\bar{A}_i}{2}$, $C_2 = \frac{\bar{A}-B_j}{2}$, $C_3 = \frac{\bar{A}_j}{2}$, $C_4 = \bar{A}_i$, $C_5 = \bar{A}_{ij}$, $C_6 = \bar{A}_j$, $C_7 = \left[\bar{R}_i(t)+\bar{A}_i\rho_{ij}\right]$, $C_8 = \left[\bar{R}_i(t)+\bar{A}\rho_{ij}\right]$, $C_9 = \left[\bar{R}_{ij}(t)+\bar{A}_j\rho_{ij}\right]$ and $C_{10} = \frac{\rho_{ij}}{2}\left[2\bar{R}_i(t)+\bar{A}\rho_{ij}\right]$. Note that $C_i \geq 0$, $\forall i$ except for $C_2$, where $C_2 \geq 0 \iff \bar{A} \geq B_j$.

**Solving the RHCP2 for Optimal Control** $(v_i^*, u_j^*, v_j^*)$**:** Based on the first step of (5.11), the optimal controls for **RHCP2** are determined by

$$(v_i^*, u_j^*, v_j^*) = \underset{(v_i, u_j, v_j)}{\arg\min}\; J_H(v_i, u_j, v_j), \tag{5.37}$$

where $(v_i, u_j, v_j) \in \mathbb{U}_1$ or $(v_i, u_j, v_j) \in \mathbb{U}_2$ as in (5.34).

**- Case 1:** $(v_i, u_j, v_j) \in \mathbb{U}_1$ in (5.34): Then, $v_j^* = 0$ and (5.37) takes the form:

$$\begin{aligned} (v_i^*, u_j^*) = \underset{(v_i, u_j)}{\arg\min}\;& J_H(v_i, u_j, 0) \\ & v_i \geq 0, \quad 0 \leq u_j \leq u_j^B(v_i), \\ & v_i + u_j \leq H - \rho_{ij}. \end{aligned} \tag{5.38}$$

The above constraints follow from (5.34) and the relationships:

$$\begin{aligned} & v_i \leq \bar{v}_i(u_j, 0) \iff v_i \leq H - (\rho_{ij} + u_j) \text{ and} \\ & u_j \leq \bar{u}_j(v_i) \iff u_j \leq u_j^B(v_i) \;\;\&\;\; u_j \leq H - (v_i + \rho_{ij}). \end{aligned}$$

Note that $u_j^B(v_i)$ (5.33) is linear in $v_i$. Prior to presenting the approach for solving (5.38), the second sub-problem of (5.37) based on $\mathbb{U}_2$ (5.34) is formulated next.

**- Case 2:** $(v_i, u_j, v_j) \in \mathbb{U}_2$ in (5.34): Then, $u_j^* = u_j^B(v_i^*)$ and (5.37) takes the form:

$$\begin{aligned} (v_i^*, v_j^*) = \underset{(v_i, v_j)}{\arg\min}\;& J_H(v_i, u_j^B(v_i), v_j) \\ & v_i \geq 0, \quad v_j \geq 0, \\ & v_i + u_j^B(v_i) + v_j \leq H - \rho_{ij}. \end{aligned} \tag{5.39}$$

The constraints in (5.39) follow from (5.34) and the relationships:

$$v_i \leq \bar{v}_i(u_j^B(v_i), v_j) \iff v_i \leq H - (\rho_{ij} + u_j^B(v_i) + v_j) \text{ and}$$
$$v_j \leq \bar{v}_j(v_i) \iff v_j \leq H - (v_i + \rho_{ij} + u_j^B(v_i)).$$

**- Combined Result:** The two optimization problems (5.38) and (5.39) belong to the class of *constrained bi-variate rational function optimization problems* (RFOPs) in (D.8) discussed in Appendix D.1. Specifically, Appendix D.1 presents a computationally efficient, analytical procedure for obtaining the globally optimal solution of such RFOPs. As will be shown in the rest of this paper, all remaining problems that need to be solved belong to this class of RFOPs.

To provide an example, note that (5.38) is a special case of (D.8) using $x := v_i$, $y := u_j$, $H(x,y) := J_H(v_i, u_j, 0)$, $\mathbf{P} := \frac{A_j}{B_j - A_j}$, $\mathbf{L} := \frac{R_j(t) + A_j\rho_{ij}}{B_j - A_j}$, $\mathbf{Q} := 1$, $\mathbf{M} := H - \rho_{ij}$ and $\mathbf{N} := \infty$. Similarly, (5.39) is also a special case of (D.8).

The main optimization problem (5.37) is solved by individually solving (5.38) and (5.39) and then simply comparing their optimal objective function values.

**Solving for Optimal (Planned) Next Destination $j^*$:** The second step of **RHCP2** (i.e., (5.12)) is to choose the optimal neighbor $j \in \mathcal{N}_i$ according to $j^* = \arg\min_{j \in \mathcal{N}_i} J_H(v_i^*, u_j^*, v_j^*)$. This step requires the objective value of the optimal solution $U_{ij}^* = [v_i^*, u_j^*, v_j^*]$ obtained for each $j \in \mathcal{N}_i$ (in (5.37)). Now, $v_i^*$ taken from $U_{ij^*}^*$ defines the inactive time that the agent should spend at current target $i$ starting from the current time $t$ until the next local event. This next event is either: (i) $[h \to v_i^*]$ or (ii) a neighbor-induced $C_j$ or $\bar{C}_j$ event for some $j \in \mathcal{N}_i$. Depending on this event, the agent will have to subsequently solve an instance of **RHCP3** or **RHCP2**, respectively.

### 5.2.3 Solution of RHCP1

An agent $a$ residing at target $i$ has to solve **RHCP1** only when an observed event is: (i) agent $a$'s arrival at target $i$, or (ii) a neighbor induced event (i.e., $C_j$ or $\bar{C}_j$ for some $j \in \mathcal{N}_i$) while $R_i(t) > 0$. Therefore, **RHCP1** involves all the decision variables $U_{ij} \triangleq [u_i, v_i, u_j, v_j]$ and $j$ included in (5.14). Upon solving **RHCP1**, the obtained $u_i^*$ gives the active time remaining to be spent at target $i$ - until the next local event occurs. The variable horizon $w$ for this case is $w = u_i + v_i + \rho_{ij} + u_j + v_j$, as in (5.13).

**Constraints:** Following the previously used notation, the maximum possible active times at targets $i$ and $j$ are respectively denoted by $u_i^B$ and $u_j^B$ where

$$u_i^B = \frac{R_i(t)}{B_i - A_i} \quad \text{and} \quad u_j^B(u_i, v_i) = \frac{R_j(t) + A_j\rho_{ij}}{B_j - A_j} + \frac{A_j}{B_j - A_j} \cdot (u_i + v_i). \tag{5.40}$$

Note that $u_j^B$ is now dependent on the control variables $u_i$ and $v_i$. Based on the same arguments as in the analysis of **RHCP2**: (i) to spend a positive inactive time at any target, the agent should spend the maximum possible active time at that target, and (ii) the variable horizon is subject to $w \leq H$. Thus, any feasible $(u_i, v_i, u_j, v_j)$ in (5.11) belongs to one of the four constraint set pairs: $(\mathbb{U}_{ik}, \mathbb{U}_{jl})$, $k, l \in \{1, 2\}$ where

$$\begin{aligned} \mathbb{U}_{i1} = &\quad \{0 \leq u_i \leq \bar{u}_i(u_j, v_j),\ v_i = 0\}, \quad \mathbb{U}_{i2} = \quad \{u_i = u_i^B,\ 0 \leq v_i \leq \bar{v}_i(u_j, v_j)\}, \\ \mathbb{U}_{j1} = &\quad \{0 \leq u_j \leq \bar{u}_j(u_i, v_i),\ v_j = 0\}, \quad \mathbb{U}_{j2} = \quad \{u_j = u_j^B(u_i, v_i),\ 0 \leq v_j \leq \bar{v}_j(u_i, v_i)\}, \end{aligned} \tag{5.41}$$

with $u_i^B$ and $u_j^B(u_i, v_i)$ given in (5.40) and

$$\begin{gathered} \bar{u}_i(u_j, v_j) = \min\{u_i^B,\ H - (\rho_{ij} + u_j + v_j)\}, \quad \bar{v}_i(u_j, v_j) = H - (u_i^B + \rho_{ij} + u_j + v_j), \\ \bar{u}_j(u_i, v_i) = \min\{u_j^B(u_i, v_i),\ H - (u_i + v_i + \rho_{ij})\}, \\ \bar{v}_j(u_i, v_i) = H - (u_i + v_i + \rho_{ij} + u_j^B(u_i, v_i)). \end{gathered} \tag{5.42}$$

These are similar to (5.19) and (5.34), but, unlike (5.19) or (5.34), each of these four limiting values are now dependent on two control decisions.

**Objective:** According to (5.14), the objective function corresponding to **RHCP1** is $J_H(U_{ij}) = J_H(X_i(t), U_{ij}; H) = \frac{1}{w}\bar{J}_i(t, t+w)$. To obtain an explicit expression for $J_H(U_{ij})$, $\bar{J}_i$ in (5.9) is decomposed as in (5.35) and the three terms $J_i$, $J_j$ and $J_m$ are evaluated for trajectories such that the agent moves from target $i$ to $j$ following decisions $[u_i, v_i, u_j, v_j]$ during the period $[t, t+w)$. With the aid of Fig. 5·5 and Theorem 5.1 it can be shown that:

$$\begin{aligned}
J_i &= \frac{u_i}{2}[2R_i(t) - (B_i - A_i)u_i] \\
&\quad + \frac{(\rho_{ij} + u_j + v_j)}{2}[2(R_i(t) - (B_i - A_i)u_i) + A_i(\rho_{ij} + u_j + v_j)], \\
J_j &= \frac{(u_i + v_i + \rho_{ij})}{2}[2R_j(t) + A_j(u_i + v_i + \rho_{ij})] \\
&\quad + \frac{u_j}{2}[2(R_j(t) + A_j(u_i + v_i + \rho_{ij})) - (B_j - A_j)u_j], \\
J_m &= \frac{(u_i + v_i + \rho_{ij} + u_j + v_j)}{2}[2R_m(t) + A_m(u_i + v_i + \rho_{ij} + u_j + v_j)].
\end{aligned}$$

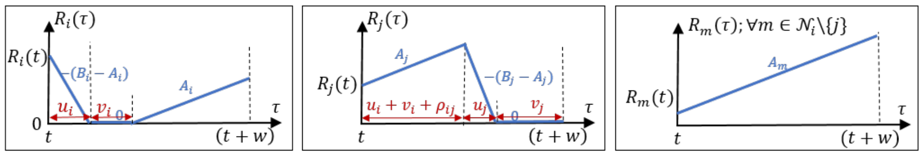


**Figure 5·5:** State trajectories of targets in $\bar{\mathcal{N}}_i$ during $[t, t+w)$.

Combining the above three results and substituting them in (5.35) gives

$$J_H(u_i, v_i, u_j, v_j) = \frac{\begin{bmatrix} C_1u_i^2 + C_2v_i^2 + C_3u_j^2 + C_4v_j^2 + C_5u_iv_i \\ + C_6u_iu_j + C_7u_iv_j + C_8v_iu_j + C_9v_iv_j + C_{10}u_jv_j \\ + C_{11}u_i + C_{12}v_i + C_{13}u_j + C_{14}v_j + C_{15} \end{bmatrix}}{u_i + v_i + \rho_{ij} + u_j + v_j}, \tag{5.43}$$

where $C_1$ through $C_{15}$ are known functions of $\bar{A}$, $A_i$, $A_j$, $B_i$, $B_j$, $\rho_{ij}$, $\bar{R}(t)$, $R_i(t)$ and $R_j(t)$ (see also (5.22)). Their specific forms are omitted here but can be found in (Welikala and Cassandras, 2020c).

**Solving the RHCP1 for Optimal Control** $(u_i^*, v_i^*, u_j^*, v_j^*)$**:** The first step of **RHCP1** (5.11) can be stated using (5.43) as

$$(u_i^*, v_i^*, u_j^*, v_j^*) = \underset{(u_i,v_i,u_j,v_j)}{\arg\min}\ J_H(u_i, v_i, u_j, v_j), \tag{5.44}$$

where $(u_i, v_i) \in \mathbb{U}_{ik}$, $(u_j, v_j) \in \mathbb{U}_{jl}$ for $k, l \in \{1, 2\}$ as in (5.41). There are four different cases for this problem depending on which of the four constraint set pairs in (5.41) is used.

**- Case 1:** $(u_i, v_i) \in \mathbb{U}_{i1}$, $(u_j, v_j) \in \mathbb{U}_{j1}$ in (5.41): Then, $v_i^* = 0$, $v_j^* = 0$ and (5.44) takes the form:

$$\begin{aligned} (u_i^*, u_j^*) = &\underset{(u_i,u_j)}{\arg\min}\ J_H(u_i, 0, u_j, 0) \\ &0 \leq u_i \leq u_i^B, \quad 0 \leq u_j \leq u_j^B(u_i, 0), \\ &u_i + u_j \leq H - \rho_{ij}. \end{aligned} \tag{5.45}$$

The above constraints follow from (5.41) and the relationships:

$$\begin{aligned} &u_i \leq \bar{u}_i(u_j, 0) \iff u_i \leq u_i^B \ \ \&\ \ u_i \leq H - (\rho_{ij} + u_j) \text{ and} \\ &u_j \leq \bar{u}_j(u_i, 0) \iff u_j \leq u_j^B(u_i, 0) \ \ \&\ \ u_j \leq H - (u_i + \rho_{ij}). \end{aligned}$$

Note that $u_j^B(u_i, 0)$ is linear and increasing with $u_i$ (see (5.40)). The remaining three cases are as follows (details can be found in (Welikala and Cassandras, 2020c)):

**- Case 2:** $(u_i, v_i) \in \mathbb{U}_{i1},\ (u_j, v_j) \in \mathbb{U}_{j2}$ in (5.41): Then, $v_i^* = 0,\ u_j^* = u_j^B(u_i^*, 0)$ and (5.44) takes the form:

$$\begin{aligned} (u_i^*, v_j^*) = &\underset{(u_i, v_j)}{\arg\min}\ J_H(u_i, 0, u_j^B(u_i, 0), v_j) \\ &0 \le u_i \le u_i^B, \quad v_j \ge 0, \\ &u_i + u_j^B(u_i, 0) + v_j \le H - \rho_{ij}. \end{aligned} \tag{5.46}$$

**- Case 3:** $(u_i, v_i) \in \mathbb{U}_{i2},\ (u_j, v_j) \in \mathbb{U}_{j1}$ in (5.41): Then, $u_i^* = u_i^B,\ v_j^* = 0$ and (5.44) takes the form:

$$\begin{aligned} (v_i^*, u_j^*) = &\underset{(v_i, u_j)}{\arg\min}\ J_H(u_i^B, v_i, u_j, 0) \\ &v_i \ge 0, \quad 0 \le u_j \le u_j^B(u_i^B, v_i), \\ &v_i + u_j \le H - (u_i^B + \rho_{ij}). \end{aligned} \tag{5.47}$$

Note that $u_j^B(u_i^B, v_i)$ is linear and increasing with $v_i$ (5.40).

**- Case 4:** $(u_i, v_i) \in \mathbb{U}_{i2},\ (u_j, v_j) \in \mathbb{U}_{j2}$ in (5.41): Then, $u_i^* = u_i^B,\ u_j^* = u_j^B(u_i^B, v_i^*)$ and (5.44) takes the form:

$$\begin{aligned} (v_i^*, v_j^*) = &\underset{(v_i, v_j)}{\arg\min}\ J_H(u_i^B, v_i, u_j^B(u_i^B, v_i), v_j) \\ &v_i \ge 0, \quad v_j \ge 0, \\ &v_i + v_j + u_j^B(u_i^B, v_i) \le H - (u_i^B + \rho_{ij}). \end{aligned} \tag{5.48}$$

**- Combined Result:** The optimization problems (5.45), (5.46), (5.47) and (5.48) belong to the same class of RFOPs in (D.8) discussed in Appendix D.1 (similar to (5.38) and (5.39)). Therefore, each of these four problems are solved exploiting the computationally efficient, analytical, and globally optimal solution presented in Appendix D.1. To provide an example, note that (5.45) conforms to (D.8) using

$x := u_i$, $y := u_j$, $H(x, y) := J_H(u_i, 0, u_j, 0)$, $\mathbf{P} := \frac{A_j}{B_j - A_j}$, $\mathbf{L} := \frac{R_j(t) + A_j\rho_{ij}}{B_j - A_j}$, $\mathbf{Q} := 1$, $\mathbf{M} := H - \rho_{ij}$, $\mathbf{N} := \frac{R_i(t)}{B_i - A_i}$.

The main optimization problem (5.44) is solved by individually solving (5.45)-(5.48) and then simply comparing their optimal objective function values.

**Solving for Optimal (Planned) Next Destination $j^*$:** The second step of **RHCP1**, same as (5.12), is to choose the optimal next neighbor $j$ according to $j^* = \underset{j \in \mathcal{N}_i}{\arg\min}\ J_H(u_i^*, v_i^*, u_j^*, v_j^*)$. This step requires the objective values corresponding to the optimal solutions $U_{ij}^* = [u_i^*, v_i^*, u_j^*, v_j^*]$ for each $j \in \mathcal{N}_i$ in (5.44). Finally, recall that $u_i^*$ within the optimal solution $U_{ij^*}^*$ defines the active time that the agent should spend at current target $i$ until the next local event occurs. This next event is either (i) $[h \to u_i^*]$ with $R_i(t + u_i^*) > 0$, (ii) $[h \to u_i^*]$ with $R_i(t + u_i^*) = 0$, or (iii) a neighbor induced $C_j$ or $\bar{C}_j$ event for some $j \in \mathcal{N}_i$ (while $R_i > 0$). Therefore, the agent will have to subsequently solve an instance of **RHCP3**, **RHCP2** or **RHCP1** respectively.

## 5.3 Controller Enhancements

There are two reasons why the proposed distributed RHC approach cannot guarantee a global minimum of (5.3). The first reason is that in order to operate in distributed fashion, the future cost estimate term $\hat{J}_H(X_i(t + H))$ in (5.4) has been omitted and a local objective function (5.9) has been defined for an agent at target $i$ which reflects the structure of (5.3) limited to the neighborhood of target $i$. In (5.9), all neighboring target states are equally weighted which does not take into account the specific neighboring topology. In particular, an optimal next-visit target $j^*$ determined by **RHCP3** favors neighbors with smaller $\rho_{ij}$ and $R_j(t)$ values. This can be alleviated by adopting different weights in the targets $j \in \mathcal{N}_i$ included in (5.9).

The second reason also stems from the distributed nature of the RHC; the information available to an agent located at target $i$ has been limited to its neighborhood

$\bar{\mathcal{N}}_i$. One can expect that performance can be improved by considering an extended neighborhood whereby additional information may become available to the agent.

In this section, these two issues are addressed. Specifically, the main focus is on improving the formulation of **RHCP3** as it involves the crucial next-visit decision $j^*$, which highly affects the agent trajectories.

### 5.3.1 Using a Weighted Local Objective in RHCP3

The local objective function decomposition in (5.20) is generalized here by introducing a weighted version of it as

$$\bar{J}_i = \alpha J_j + (1-\alpha) \sum_{m \in \bar{\mathcal{N}}_i \backslash \{j\}} J_m, \tag{5.49}$$

where $\alpha \in [0,1]$. This approach is more effective as it can emphasize the contribution to the global cost by the "neglected neighbor targets" $m \in \bar{\mathcal{N}}_i \backslash \{j\}$ due to the choice of target $j \in \mathcal{N}_i$ as the next-visit. The modified RHC approach that uses (5.49) is referred to as the "RHC$^\alpha$ method". It should be noted that this modification has no significant effect on the theoretical results presented in the previous sections.

Regarding desirable values of the weight $\alpha$, $\alpha < 0.5$ has been found to provide high performing solutions in experiments. In order to extend the parameter-free nature of the original RHC method to this RHC$^\alpha$ method, $\alpha = \frac{1}{|\bar{\mathcal{N}}_i|^2}$ is used as a nominal choice, so as to reduce the importance of target $j$ based on the size of the neighborhood of target $i$.

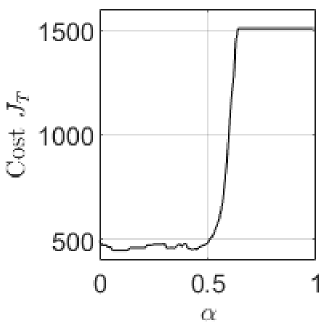
**Lemma 5.3.** *If $\alpha = 0$ is used in RHC$^\alpha$, the optimal next-visit target $j^*$ given by* ***RHCP3*** *(i.e., the solution to* (5.12)*) is*

$$j^* = \operatorname*{arg\,min}_{j \in \bar{\mathcal{N}}_i} \left[ (2\bar{R}(t) + \bar{A}\rho_{ij}) - (2R_j(t) + A_j \rho_{ij}) \right]. \tag{5.50}$$

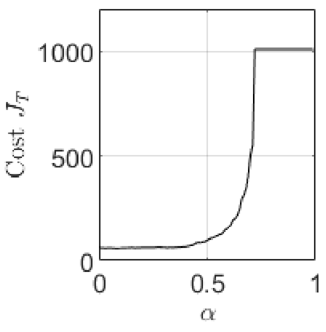
*Proof.* See Appendix A.5.5. □

Note that the first and second terms in (5.50) approximate the contribution to the main objective (5.3) during the travel-time $\rho_{ij}$ of the neighborhood and of target $j$ respectively. This is an important result (even if it is valid only under $\alpha = 0$) as it provides a direct, simple and intuitive approach to obtain the next-visit target decision $j^*$ (skipping (5.11)) for **RHCP3**.

Finally, to provide some intuition regarding how the choice of $\alpha$ affects the PMN problem performance, Fig 5·6 shows examples of how performance varies with $\alpha$ in two specific PMN problems. It can be seen that $\alpha = 0$ is sometimes directly the optimal choice while in other cases there may be a particular $\alpha < 0.5$ which provides the optimal performance.



(a) Single-Agent Case in Fig. 5·10



(b) Multi-Agent Case in Fig. 5·13

**Figure 5·6:** Variation of $J_T$ in (5.3) with $\alpha$ in (5.49).

## 5.3.2 Extending RHCP3 to a Two-Target Look Ahead

In accordance with the goal of the RHC being decentralized, **RHCP3** limits feasible agent trajectories to a one-target lookahead $j \in \mathcal{N}_i$ ahead of target $i$. Therefore, an obvious extension expected to provide improvements is to consider agent trajectories two targets ahead of $i$, assuming such information can be provided to target $i$. This is achieved by extending the associated planning horizon of **RHCP3** as shown in Fig. 5·7 so that it includes an extra target visit to $k \in \mathcal{N}_j$ beyond vising target $j \in \mathcal{N}_i$.

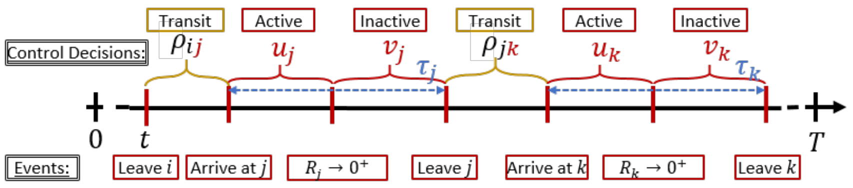


**Figure 5·7:** Extended planning timeline for **RHCP3**.

In this case, the interested real-valued and discrete decision variables become $U_{ijk} = [u_j, v_j, u_k, v_k]$ and $\{j,k\}$ respectively. The variable horizon $w$ defined in (5.13) becomes $w = \rho_{ij} + u_j + v + j + \rho_{jk} + u_k + v_k$. To obtain the optimal values of these decision variables, first, the concepts of neighborhood $\bar{\mathcal{N}}_i$ (5.2), local objective $\bar{J}_i$ (5.9) and local state $X_i(t)$ are extended respectively as $\tilde{\mathcal{N}}_i$, $\tilde{J}_i$, and $\tilde{X}_i(t)$ where

$$\tilde{\mathcal{N}}_i = \cup_{j\in\bar{\mathcal{N}}_i}\mathcal{N}_j, \quad \tilde{J}_i = \sum_{m\in\tilde{\mathcal{N}}_i} J_m, \quad \tilde{X}_i(t) = \{R_m(t); m \in \tilde{\mathcal{N}}_i\}.$$

Now, the extended **RHCP3** (by modifying (5.11), (5.12) and (5.14)), takes the form:

$$U_{ijk}^* = \underset{U_{ijk}\in\mathbb{U}}{\arg\min}\ J_H(\tilde{X}_i(t), U_{ijk}; H);\ \forall j \in \mathcal{N}_i, \forall k \in \mathcal{N}_j, \tag{5.51}$$

$$\{j^*, k^*\} = \underset{j\in\mathcal{N}_i,\ k\in\mathcal{N}_j}{\arg\min}\ J_H(\tilde{X}_i(t), U_{ijk}^*; H) \text{ with} \tag{5.52}$$

$$\begin{gathered} J_H(\tilde{X}_i(t), U_{ijk}; H) = \frac{1}{w}\tilde{J}_i(t, t+w) \text{ and} \\ \mathbb{U} = \{U : U \in \mathbb{R}^4, U \geq 0, |U| + \rho_{ij} + \rho_{jk} \leq H\}. \end{gathered} \tag{5.53}$$

This problem in (5.51) shares many similarities with **RHCP1** in (5.11) and each of its sub-problems belongs to the same class of RFOPs in (D.8) discussed in Appendix D.1, similar to those of **RHCP1** and **RHCP2**. Hence, the details of how (5.51) and (5.52) are solved are omitted (but can be found in (Welikala and Cassandras, 2020c)).

This extended RHC method is referred to as the "Ex-RHC method". Similar to (5.49), the neighborhood objective function decomposition $\tilde{J}_i$ in (5.53) can also be modified from:

$$\tilde{J}_i = J_j + J_k + \sum_{m \in \tilde{\mathcal{N}}_i \backslash \{j,k\}} J_m, \tag{5.54}$$

to incorporate weight factors $\alpha, \beta \in [0, 1]$ as:

$$\tilde{J}_i = \alpha J_j + \beta J_k + (1 - \alpha - \beta) \sum_{m \in \tilde{\mathcal{N}}_i \backslash \{j,k\}} J_m. \tag{5.55}$$

The motivation behind adopting (5.55) is to emphasize the contribution of the neglected neighborhood targets $m \in \tilde{\mathcal{N}}_i \backslash \{j, k\}$ and by the target $k$ to the global cost. Therefore, $\alpha < \frac{1}{3}$ and $\beta < \frac{2}{3} - \alpha$ are preferred and $\alpha = \frac{1}{|\mathcal{N}_i|^2}$ and $\beta = \frac{1}{|\mathcal{N}_i|}$ are selected as the nominal choices. By analogy to RHC$^{\alpha}$, this modified approach is referred to as the "Ex-RHC$^{\alpha\beta}$ method".

Aside from the obvious increment in computational requirements, one clear disadvantage of the Ex-RHC (or Ex-RHC$^{\alpha\beta}$) method is that agents now require more information to make their next-visit target $j^*$ decisions, thus compromises the distributed nature of the solution. Even though it is reasonable to expect that the payoff of such a compromise is a considerable performance improvement, this is far from evident in the numerical examples shown in Section 5.4. One reason may be the substantial errors in estimating $R_k(t)$ trajectories (required to evaluate $J_k$ in (5.54) (or (5.55))) when there are multiple agents and when target $k$ is located far from target $i$. These errors indirectly and negatively affect the crucial $j^*$ decision. However, from simulation results, it was found that the Ex-RHC (or Ex-RHC$^{\alpha\beta}$) method generally performs better when (i) the number of both agents and targets are relatively low, and (ii) travel-times in the graph are relatively short.

## 5.4 Simulation Results

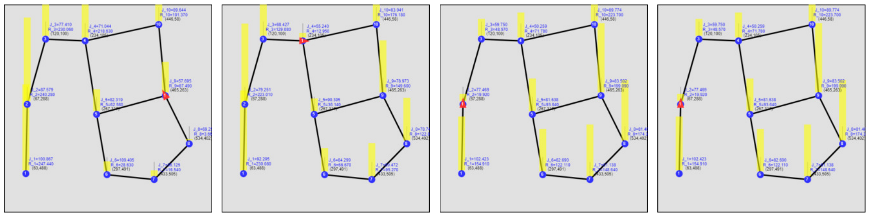
In this section, the performance measured through $J_T$ in (5.3) is compared over several different PMN problem configurations using: (i) The IPA-TCP method (Zhou et al., 2019) (ii) The RHC method proposed in Section 5.2, (iii) The RHC$^{\alpha}$ method (Section 5.3.1), and (iv) The Ex-RHC$^{\alpha\beta}$ method (Section 5.3.2). These four methods are distributed on-line solutions in contrast to the centralized off-line solution proposed in the previous chapter. Similar to previous chapters, each of these control solutions has been implemented in a JavaScript based simulator, which is made available at `http://www.bu.edu/codes/simulations/shiran27/PersistentMonitoring/`. Readers are invited to reproduce the reported results and also to try new problem configurations using the developed interactive simulator.

In particular, this section considers three PMN problem configurations with a single agent as shown in Figs. 5·8-5·10 and with multiple agents as shown in Figs. 5·11-5·13. Similar to the reported results in the previous chapter, blue circles represent the targets, while black lines represent trajectory segments that agents can use to travel between targets. Red triangles and yellow vertical bars indicate the agent locations and the target uncertainty levels, respectively. Moreover, since both $s_a(t)$ and $R_i(t)$ are time-varying, the figures show only their state at time $t = T$.

In each problem configuration, the target parameters were chosen as $A_i = 1,\ B_i = 10$ and $R_i(0) = 0.5,\ \forall i \in \mathcal{T}$ and all the targets were placed inside a $600 \times 600$ mission space. The overall time period is $T = 500$. Each agent is assumed to follow first-order dynamics (similar to (Zhou et al., 2019)) with a maximum speed of $V_{ij} = 50$ units per second on each trajectory segment $(i, j) \in \mathcal{E}$. The initial locations of the agents were chosen such that they are uniformly distributed among the targets at time $t = 0$ (i.e., $s_a(0) = Y_i$ with $i = 1 + (a - 1) \times \text{round}(M/N)$). The (non-critical) upper-bound for each planning horizon $w$ was chosen as $H = T/2 = 250$ for the three RHC approaches and $\alpha = \frac{1}{|\mathcal{N}_i|^2}$, $\beta = \frac{1}{|\mathcal{N}_i|}$ were used in the RHC$^\alpha$ and Ex-RHC$^{\alpha\beta}$ methods.

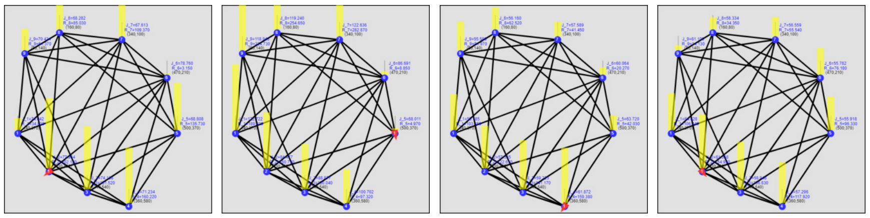
**Table 5.1:** Summary of obtained results.

| $J_T$ in (5.3) | Single-Agent Simulation Examples | | | Multi-Agent Simulation Examples | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 |
| IPA-TCP | 831.3 | 651.3 | 497.9 | 270.2 | 274.0 | 201.3 |
| RHC | 791.1 | 912.3 | 490.4 | 105.5 | 114.1 | 97.2 |
| RHC$^\alpha$ | **790.1** | **527.7** | 464.8 | 96.6 | **63.7** | **60.1** |
| Ex-RHC$^{\alpha\beta}$ | 790.1 | 529.7 | **449.5** | **95.7** | 75.0 | 70.2 |



**(a)** IPA-TCP: $J_T = 831.3$ **(b)** RHC: $J_T = 791.1$ **(c)** RHC$^\alpha$: $J_T = \mathbf{790.1}$ **(d)** Ex-RHC$^{\alpha\beta}$: $J_T = 790.1$

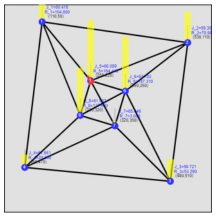
**Figure 5·8:** Single-agent simulation example 1 (SASE1).

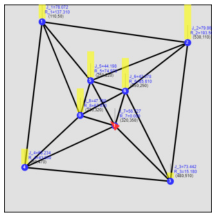


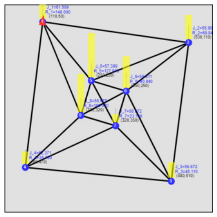
**(a)** IPA-TCP: $J_T = 651.3$ **(b)** RHC: $J_T = 912.2$ **(c)** RHC$^\alpha$: $J_T = \mathbf{527.7}$ **(d)** Ex-RHC$^{\alpha\beta}$: $J_T = 529.7$

**Figure 5·9:** Single-agent simulation example 2 (SASE2).

Each sub-figure caption in Figs. 5·8-5·13 provides the cost value $J_T$ in (5.3) observed under each controller (i.e., either IPA-TCP, RHC, RHC$^\alpha$ or Ex-RHC$^{\alpha\beta}$). These cost values are summarized in Tab. 5.1. From the observed results, note that the proposed RHC method has performed considerably better (on average 57.0% better) than the IPA-TCP method for multi-agent problem configurations. For single-agent problem configurations, both methods have performed equally except for SASE3. The proposed RHC$^\alpha$ approach further improves these performances compared to the


**(a)** IPA-TCP: $J_T = 497.9$


**(b)** RHC: $J_T = 490.4$


**(c)** RHC$^\alpha$: $J_T = 464.8$

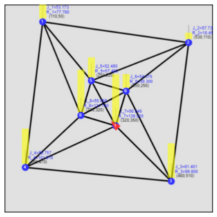
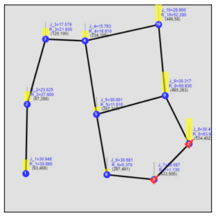
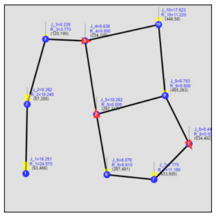
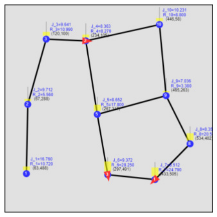

**(d)** Ex-RHC$^{\alpha\beta}$: $J_T = \mathbf{449.5}$

**Figure 5·10:** Single-agent simulation example 3 (SASE3).


**(a)** IPA-TCP: $J_T = 270.2$


**(b)** RHC: $J_T = 105.5$


**(c)** RHC$^\alpha$: $J_T = 96.6$

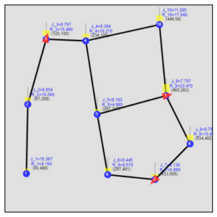
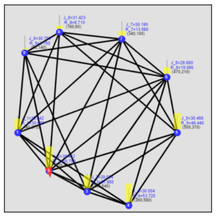
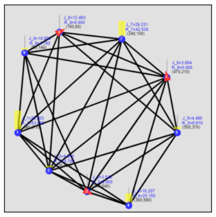
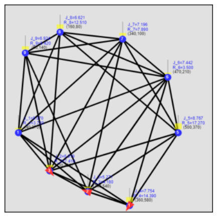

**(d)** Ex-RHC$^{\alpha\beta}$: $J_T = \mathbf{95.7}$

**Figure 5·11:** Multi-agent simulation example 1 (MASE1).


**(a)** IPA-TCP: $J_T = 274.0$


**(b)** RHC: $J_T = 114.1$


**(c)** RHC$^\alpha$: $J_T = \mathbf{63.7}$

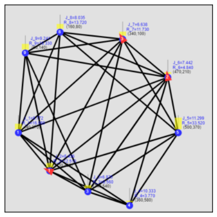
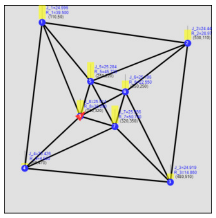
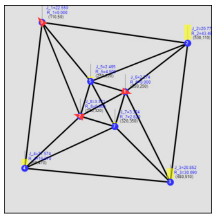
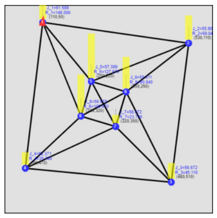

**(d)** Ex-RHC$^{\alpha\beta}$: $J_T = 75.0$

**Figure 5·12:** Multi-agent simulation example 2 (MASE2).


**(a)** IPA-TCP: $J_T = 201.3$


**(b)** RHC: $J_T = 97.2$


**(c)** RHC$^\alpha$: $J_T = \mathbf{60.1}$

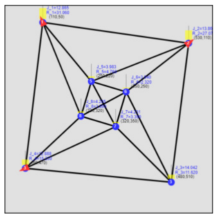

**(d)** Ex-RHC$^{\alpha\beta}$: $J_T = 70.2$

**Figure 5·13:** Multi-agent simulation example 3 (MASE3).

IPA-TCP method by an average of 70.4% for multi-agent situations and by 10.2% for single-agent situations. On the other hand, the proposed Ex-RHC$^{\alpha\beta}$ method provides on average 67.4% and 11.1% improvements respectively for multi-agent and single-agent cases compared to the IPA-TCP method. In light of the fact that Ex-RHC$^{\alpha\beta}$ compromises the distributed nature of the original RHC, all evidence points to the conclusion that there is no benefit to this extension for most network topologies.

Finally, in order to determine the RHC version (out of RHC$^{\alpha}$ and Ex-RHC$^{\alpha\beta}$) with superior robustness properties, an extensive empirical study has been carried out that investigates the robustness to the randomness in different aspects of the persistent monitoring system (e.g., system parameters, state dynamics or state information shared with neighbors). This study has also lead to the conclusion that RHC$^{\alpha}$ method outperforms the Ex-RHC$^{\alpha\beta}$ method in terms of its robustness properties (details are omitted here but can be found (Welikala and Cassandras, 2020c)).

**Table 5.2:** (From Tab. 4.1) A performance comparison between PMN solutions proposed in (Zhou et al., 2019), Chapter 4 and Chapter 5. The average percentage improvement achieved by the latter two solutions compared to (Zhou et al., 2019) are: 44.34% and 42.57, respectively.

| Persistent Monitoring Objective $J_T$ | Single-Agent Simulation Examples | | | Multi-Agent Simulation Examples | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 1 | 2 | 3 | 4 |
| TCP solution proposed in (Zhou et al., 2019) (Distributed) | 129.2 | 651.3 | 497.9 | 270.2 | 91.7 | 274.0 | 201.3 |
| TCP solution proposed in Chapter 4 (Centralized) | 114.6 | 567.0 | 449.5 | 90.9 | 35.1 | 59.5 | 59.8 |
| RHC solution proposed in Chapter 5 (Distributed) | 121.4 | 527.7 | 464.8 | 96.6 | 40.4 | 64.9 | 60.7 |

## 5.5 Summary

A distributed receding horizon control based solution was developed for the class of persistent monitoring on network problems. This controller exploits the event-driven nature of the underlying PMN system to reduce the computational complexity significantly. Further, it automatically optimizes its planning horizon length, thus making the controller parameter-free. Furthermore, explicit globally optimal solutions were derived for every distributed optimization problem encountered at each event where the receding horizon controller is invoked. As a result of these contributions, compared to the centralized parametric controller proposed in the previous chapter, this distributed receding horizon controller has several practical advantages such as being: gradient-free, parameter-free, initialization-free, event-driven, distributed, on-line as well as computationally efficient. Understandably, these advantages come at the cost of a small performance loss compared to the centralized parametric control solution proposed in the previous chapter (see Tab. 5.2). Nevertheless, as shown in the previous section (and also in Tab. 5.2), compared to the distributed parametric control solution (Zhou et al., 2019) (which also uses a parameter learning stage), the proposed distributed receding horizon control solution provides significant improvements (without any learning stages).

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

This dissertation focused on addressing the issue of local optima arising in control and optimization of cooperative multi-agent systems with inherent non-convexities. In particular, several systematic techniques were developed to facilitate either escaping or avoiding poor local optima while in search of better optima.

In Chapter 2, for a large class of multi-agent optimization problems, a systematic distributed approach to escape local optima using the concept of "boosting functions" was proposed. First, a formal distributed boosting scheme was proposed together with a variable step size scheme to guarantee its convergence. Then, a generic boosting function form that can be adopted across multiple applications was proposed. Next, to provide more intuition into the process of designing boosting functions, for a class of coverage problems, two new boosting function families were designed while outlining the underlying motivations behind each step of the design process. Finally, the same class of coverage problems was used to validate the proposed boosting functions approach, i.e., to validate the effectiveness of the proposed: distributed boosting scheme, variable step size scheme and boosting function families. Obtained theoretical and empirical results lead to the conclusion that when addressing non-convex multi-agent optimization problems, boosting functions can enable a systematic escape from local optima that limits the conventional gradient-based methods so as to achieve superior performance levels without significantly affecting the involved computational cost.

For cooperative multi-agent optimization problems, as opposed to the distributed on-line approach that can escape poor local optima proposed in Chapter 2, a centralized off-line approach that can avoid poor local optima was explored in Chapter 3. In particular, the use of greedy algorithms in generating effective initial conditions (possibly for subsequent gradient processes) was explored. This study was motivated by the fact that such greedy methods are computationally cheap and can often provide performance bound guarantees exploiting submodularity properties of the problem. First, considering the class of submodular maximization problems, several established performance bounds were reviewed. Next, for the same class of problems, computationally efficient two new performance bounds were proposed. Then, a class of coverage problems (the same as that used in Chapter 2) was modeled as a class of submodular maximization problems so as to study the effectiveness of different performance bounds discussed. Finally, numerical results were provided to highlight the achieved improvements compared to state of the art in terms of performance bounds. This study leads to the conclusion that greedy methods, when coupled with a tight performance bound computation technique, can provide efficient, reasonable as well as reliable solutions to difficult non-convex multi-agent optimization problems.

Inspired by the efficacy of greedy solutions obtained for multi-agent optimization problems seen in Chapter 3, a greedy approach was proposed for a class of multi-agent control problems in Chapter 4. In particular, the class of persistent monitoring on networks problems was considered where a team of agents is deployed to monitor a set of targets interconnected according to a network topology aiming to minimize a measure of overall target state. First, to address this control problem, a class of distributed threshold-based parametric controllers was adopted where IPA is used to determine the optimal threshold parameters in an on-line manner using gradient descent. Next, to address the associated non-convexities, a systematic greedy threshold parameter initialization scheme was proposed. In this endeavor, asymptotic analysis of the PMN system together with graph clustering and several other combinatorial optimization algorithms were utilized. Finally, simulation results were provided to show that such systematically chosen initial threshold parameters while having significantly improved performance levels compared to the state of the art, are almost immediately (locally) optimal or quickly lead to optimal values. The findings of this chapter lead to the conclusion that even when addressing multi-agent control problems, a meticulously designed greedy method that exploits underlying structural properties of the interested problem can achieve high performing solutions.

Note that in some multi-agent control problems, it might not be feasible to have a centralized off-line initialization stage or an on-line learning stage due to various limitations in the problem setup. Considering such a limited scenario, for the class of PMN problems, a receding horizon control solution was proposed in Chapter 5 as an alternative to the parametric control solution proposed in Chapter 4. First, the event-driven nature of the PMN system was exploited to formulate a receding horizon control problem (over a planning horizon ahead) for each agent to solve upon each event of interest observed in its trajectory. Next, the determination of optimal planning horizon length was included in this RHCP via introducing the concept of "variable horizon." Then, explicit globally optimal solutions were derived for every possible RHCP form that an agent may face in its trajectory. As a result of these contributions, the proposed receding horizon controller has several practical advantages such as being: gradient-free, parameter-free, initialization-free, event-driven, distributed, on-line as well as computationally efficient. Finally, simulation results were provided to highlight the improvements achieved compared to an existing distributed on-line PMN solution. It is easy to see that this receding horizon control solution shares some similarities with the earlier proposed boosting functions approach (in Chapter 2) as both are distributed on-line processes that intermittently motivate agents to search for the optimal agent behaviors/states.

## 6.2 Ongoing and Future Research: Multi-Agent Optimization

**Applications of Boosting Functions Approach:** The effectiveness of the proposed boosting functions approach in Chapter 2 has mainly been validated using a class of coverage problems. However, this class of coverage problems can be seen as a much broader class of resource allocation problems (which also includes a particular class of consensus problems). Nevertheless, finding other different classes of multi-agent optimization or control problems where this boosting functions approach can be successfully adopted to overcome the issue of local optima would be an interesting future research direction. While such a study will lead to improved solutions, it will also enrich the existing collection of boosting function design ideas. To this end, Fig. 6·1 shows a preliminary result obtained from a study where the boosting functions approach has been adopted to address the class of PMN problems. It shows that a single boosting session has improved the performance by around 50% in the considered PMN example.

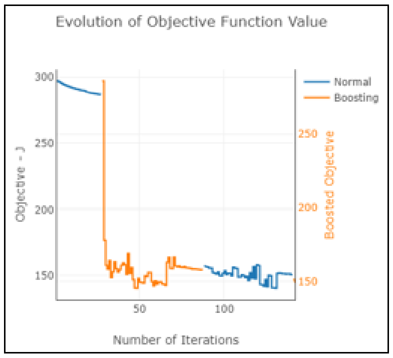


**Figure 6·1:** The impact of a boosting session during an on-line threshold parameter tuning phase in a PMN application.

**Distributed On-Line Greedy Algorithms:** One of the contributions of Chapter 3 is that it shows the greedy algorithm in Alg. 3.2 can be equivalently executed in a distributed manner (with the modification in (3.29)). However, this distributed solution requires agents to be deployed sequentially. Hence the resulting greedy algorithm can be seen as an off-line solution. Therefore, an interesting future research direction is to develop a fully distributed on-line greedy algorithm. In such an endeavor, the established theoretical results in Theorems 3.6, 3.7 and C.1 can be expected to be useful - specifically when developing update laws for the agents and performance bounds.

**Greedy - Gradient Solutions:** Intuitively the centralized off-line greedy solution proposed in Chapter 3 and the distributed on-line gradient-based solution proposed in Chapter 2 can be combined to address hard multi-agent optimization problems. For example, similar greedy-gradient approaches have been used in (Sun et al., 2017b; Sun et al., 2020). In such situations, one arising challenge is to impose performance bounds on intermediate or terminal solutions (rather than on the initial greedy solutions). This is an interesting research direction where a preliminary solution has already been proposed in (Sun et al., 2020).

## 6.3 Ongoing and Future Research: Persistent Monitoring on Networks

### 6.3.1 RHC with Reinforcement Learning

In the proposed RHC solution in Chapter 5, the future cost term of each RHCP (i.e., the $\hat{J}_H(X_i(t+H))$ term in (5.4)) was omitted. The potential myopic agent behaviors due to this omission were prevented by optimizing the planning horizon length and introducing controller enhancements. Motivated by this, an interesting future research direction would be to include this future cost term back in the RHCP and use policy iterations to approximate the exact form of it.

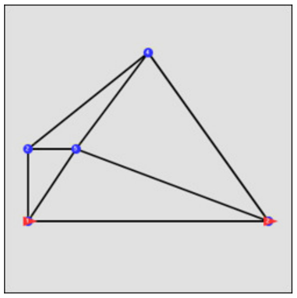
**A Preliminary Example:** In a preliminary example,

$$\hat{G}^a_{ij}(t;\,\Theta) \triangleq \theta_0 t^{-1} + \theta_1 + \theta_2 t + \theta_3 t^2,$$

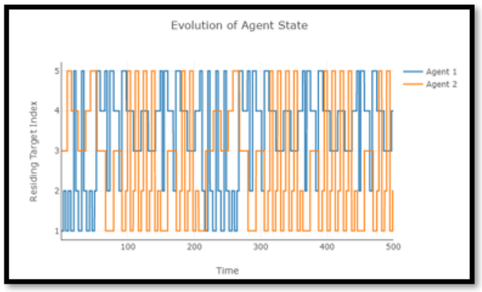
was used as the candidate future cost function - for an agent $a$ at target $i$ solving a RHCP at time $t$ aiming to visit a target $j \in \mathcal{N}_i$. Upon defining an appropriate reward function and by running several policy iterations (considering the SASE1 shown in Fig. 6·2(a)), the set of parameters $\Theta = [\theta_0, \theta_1, \theta_2, \theta_3]$ for all $i, j \in \mathcal{T}, a \in \mathcal{A}$ were learned. As respectively shown in Figs. 6·2(d) and (b), the resulting "Reinforced" Event-Driven Receding Horizon Control (RRHC) method provides improved results compared to the RHC$^\alpha$ method proposed in Chapter 5. Note also that this RRHC method generates similar results to those of the greedy method proposed in Chapter 4 (both in terms of the performance and agent trajectory).

### 6.3.2 The Use of Energy-Aware Second-Order Agents

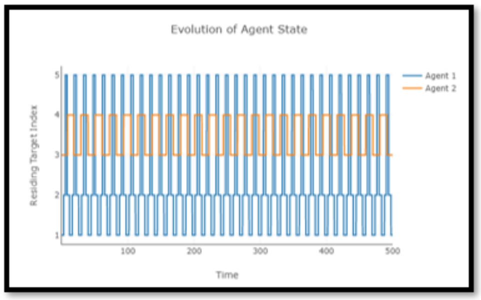
In the PMN problem setup considered in Chapters 4 and 5, each deployed agent has been assumed to follow a first-order dynamic model. Moreover, agent energy consumption associated with the motion has been neglected from the main objective function (thus, energy-agnostic). Therefore, an obvious way to improve this PMN problem setup is by allowing each agent to follow a second-order dynamic model (governed by acceleration) and by incorporating agent energy consumption into the main objective function. In particular, in the ongoing research (Welikala and Cassandras, 2021c; Welikala and Cassandras, 2021d), an agent $a \in \mathcal{A}$ is assumed to follow



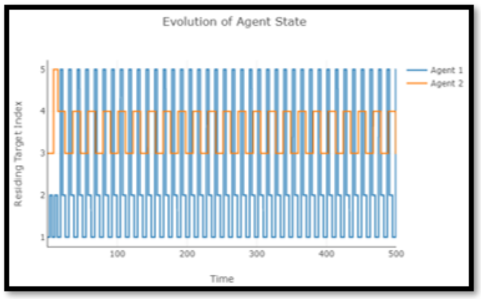
**(a)** SASE1
See Fig. 4·15



**(b)** RHC$^\alpha$
$J_T = 40.36$



**(c)** Greedy:
$J_T = \mathbf{35.13}$



**(d)** RRHC:
$J_T = 35.50$

**Figure 6·2:** A comparison of agent trajectories obtained for (a) SASE1 (see also Fig. 4·15), under different agent control methods: (b) RHC$^\alpha$, (c) Greedy (TCP) and (d) RRHC

the second-order unicycle dynamics given by

$$\dot{s}_a(t) = v_a(t)\begin{bmatrix}\cos(\theta_a(t))\\ \sin(\theta_a(t))\end{bmatrix},\ \dot{v}_a(t) = u_a(t),\ \dot{\theta}_a(t) = w_a(t), \tag{6.1}$$

where $v_a(t), u_a(t), \theta_a(t)$ and $w_a(t)$ represent the agent tangential velocity, tangential acceleration, orientation and angular velocity, respectively. Moreover, a composite objective $J_T$ of the total energy spent and the mean system uncertainty (previously, the main objective (4.2),(5.3)) is considered where

$$J_T \triangleq \alpha \int_0^T \sum_{a\in\mathcal{A}} u_a^2(t)\,dt + \frac{1}{T}\int_0^T \sum_{i\in\mathcal{T}} R_i(t)\,dt, \tag{6.2}$$

($\alpha$ is a weight factor). Under these modifications, the new challenge is to determine the optimal agent controls (and thus optimal travel-times) over trajectory segments when transitioning from one target to the next.

As described below, the ongoing research suggests that this advanced PMN problem can be solved in two different ways using the respective concepts developed in Chapters 4 and 5 of this thesis.

**Centralized Off-Line Greedy Solution:** Considering an agent traversing a target-cycle as shown in Fig. 6·3 and extending the asymptotic analysis presented in Chapter 4, it can be shown that this advanced PMN problem translates into a discrete time optimal control problem (under linear dynamics and a non-linear stage cost):

$$\begin{aligned} \{\bar{\rho}_k^* : k \in \mathcal{N}\} = &\underset{\{\bar{\rho}_k : k \in \mathcal{N}\}}{\text{argmin}} \sum_{k \in \mathcal{N}} J_k(\bar{R}_{k-1}, \bar{\rho}_k), \\ &\text{subject to}: \ \bar{R}_k = \mathbf{A}\bar{R}_{k-1} + \mathbf{B}\bar{\rho}_k, \end{aligned} \tag{6.3}$$

where $\bar{\rho}_k^*$ is the optimal set of travel-times to be used in the $k^{\text{th}}$ tour on the target-cycle (definitions of the remaining symbols are omitted for brevity). Considering a steady state scenario (as did in Chapter 4), it can be shown that:

$$\rho_i^* = \left[\frac{36\alpha}{k_\Xi \sum_{j \in \Xi} \sqrt{y_j}}\right]^{\frac{1}{5}} \sqrt{y_i}, \tag{6.4}$$

where $\rho_i^*$ is the $i^{\text{th}}$ component of $\lim_{k\to\infty} \bar{\rho}_k^*$, $y_i$ is the $i^{\text{th}}$ trajectory segment's length in the target-cycle and $k_\Xi$ is a fixed constant specific to the target-cycle. This closed form steady-state solution can now be used to create a metric (analogous to the $J_{ss}(\Xi)$ metric established in Theorem 4.1) to assess and compare different target-cycles. With such a metric, it is clear that an algorithm analogous to Alg. 4.1 can be used to completely solve this advanced PMN problem in a centralized off-line stage.

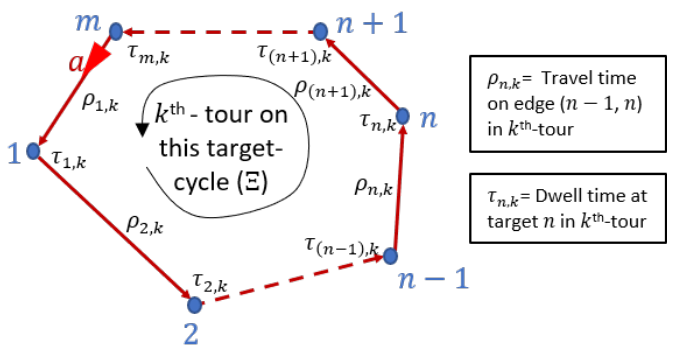

$m$
$n+1$
$a$
$\tau_{m,k}$
$\tau_{(n+1),k}$
$\rho_{(n+1),k}$
$\rho_{1,k}$
$k^{th}$ - tour on this target-cycle ($\Xi$)
$\tau_{n,k}$
$n$
$1$
$\tau_{1,k}$
$\rho_{n,k}$
$\rho_{2,k}$
$\tau_{(n-1),k}$
$\tau_{2,k}$
$n-1$
$2$
$\rho_{n,k}$= Travel time on edge $(n-1, n)$ in $k^{th}$-tour
$\tau_{n,k}$= Dwell time at target $n$ in $k^{th}$-tour

**Figure 6·3:** The target cycle $\Xi$

**Distributed On-Line RHC Solution:** As an alternative, the ongoing research (Welikala and Cassandras, 2021c; Welikala and Cassandras, 2021d) has also shown that the proposed distributed on-line RHC solution in Chapter 5 can be extended to address this advanced PMN problem in a distributed on-line manner. In particular, each RHCP is modified to include an optimal control problem that determines the optimal agent controls and the optimal travel-time over the trajectory segment corresponding to the considered RHCP.

## 6.4 Ongoing and Future Research: Distributed Estimation

In this thesis, the techniques proposed to overcome the issue of local optima in multi-agent control problems (i.e., Greedy and RHC techniques, proposed respectively in Chapters 4 and 5) have been established purely based on the class of persistent monitoring on networks problems. However, as shown in Section 6.3.2, these techniques can still be applied to the advanced version of the PMN problem (i.e., when energy-aware second-order agents are deployed). Therefore, a reasonable question to raise is: What other multi-agent control applications are there in which these developed Greedy and RHC techniques can be applied?

To answer this question, the ongoing research of this thesis considers the multi-agent distributed estimation problem over a network system (Welikala and Cassandras, 2021a; Welikala and Cassandras, 2020b). While this particular problem shares several structural similarities with the PMN problem, the target dynamics (that the agents can control) and the objective function of interest take completely different forms. In particular, here, an agent can control (by residing or not residing at target $i$) the target state estimation error covariance matrix $\Omega_i(t)$ according to the dynamics

$$\dot{\Omega}_i(t) = A_i\Omega_i(t) + \Omega_i(t)A_i' + Q_i - \eta_i(t)\Omega_i(t)G_i\Omega_i(t), \tag{6.5}$$

where $\eta_i(t) = \mathbf{1}\{\text{An agent resides at the target } i \text{ at time } t\}$ (analogous to $N_i(t)$ in (4.1) and (5.1)) and $A_i, Q_i, G_i$ are known matrices at target $i$ (analogous to the target parameters $A_i, B_i$ in (4.1) and (5.1)). The interested objective function takes the form

$$J_T = \frac{1}{T}\int_0^T \sum_{i\in\mathcal{T}} \operatorname{tr}(\Omega_i(t))dt. \tag{6.6}$$

The ongoing research suggests that this class of distributed estimation problems can be solved in two different ways using the respective concepts proposed in Chapters 4 and 5 of this thesis (see also (Pinto et al., 2020b) and (Welikala and Cassandras, 2021a; Welikala and Cassandras, 2020b), respectively for details).

In conclusion, these latest developments suggest that the concepts proposed in Chapters 4 and 5 of this thesis are not limited to PMN problems but can also be applied to address a much broader class of multi-agent control problems.

# Appendix A

# Proofs

## A.2 From Chapter 2

### A.2.1 Proof of Lemma 2.1

Consider a function $g = -f : \mathbb{R}^n \to \mathbb{R}$. Then, the Lipschitz continuity constant of $\nabla g$ will also be $L$. The usual descent lemma (Bertsekas, 2016) now can be applied to the function $g$ (to compare $g(x+y)$ and $g(x)$). Then, using $g = -f$, $\forall x, y \in \mathbb{R}^n$, $-g(x+y) \geq -g(x) - y^T \nabla g(x) - \frac{L}{2}\|y\|^2$, and the result follows.

### A.2.2 Proof of Lemma 2.3

In (2.15), add and subtract $\sum_{l \in B_j - \{i\}} \Delta_{jl,k}$ to the inner terms of the main summation. Then, using the definition (2.14), the expression in (2.19) is obtained. To prove the second part, note that the first inner term of the main summation of (2.19) (i.e., $\tilde{\Delta}_{j,k}$) is always positive under the optimal step size given in (2.17). Next, consider the net effect of the second inner term of $Q_{i,k}$, denoted by $Q'_{i,k}$, where

$$Q'_{i,k} = Q_{i,k} - \sum_{j \in B_i} \tilde{\Delta}_{j,k} = \sum_{j \in B_i} \sum_{l \in B_j - \{i\}} \left[\Delta_{lj,k} - \Delta_{jl,k}\right].$$

Using the fact that $\Delta_{lj,k} - \Delta_{jl,k} = 0$ when $l = j$, a dummy term can be added into the inner summation to get

$$Q'_{i,k} = \sum_{j \in B_i} \sum_{l \in \bar{B}_j - \{i\}} \left[\Delta_{lj,k} - \Delta_{jl,k}\right] = \sum_{j \in B_i} \sum_{l \in B_i} \left[\Delta_{lj,k} - \Delta_{jl,k}\right],$$

where the last step follows from the assumption $B_i = \bar{B}_j - \{i\}$. $Q'_{i,k} = 0$ is evident from observing that the two running variables $l, j$ in the summations above are interchangeable. This implies that under (2.17), $Q_{i,k} = \sum_{j \in B_i} \tilde{\Delta}_{j,k} > 0$.

### A.2.3 Proof of Theorem 2.1

By Assumption 2.3, a $T_i$ value can be defined for $\tilde{Q}_{i,k}$ at each $k$. Consider a sequence of consecutive discrete update instants $\{k_1+1, \ldots, k'_1\}$ (for short, the notation $(k_1, k'_1]$ is used), where, $T_i = k'_1 - k_1$ is associated with $\tilde{Q}_{i,k'_1}$ and $T_i > k - k_1$ applies to all $\tilde{Q}_{i,k}$, $k \in (k_1, k'_1 - 1]$. This means $0 < \sum_{k=k_1+1}^{k'_1} Q_{i,k}$ and $0 \geq \sum_{k=k_1+1}^{l} Q_{i,k}$, $\forall l \in (k_1, k'_1 - 1]$. In addition, by Lemma 2.2, $0 < \tilde{\Delta}^*_{i,k}$ $\forall k \in (k_1, k'_1]$. Thus, $0 < \sum_{k=k_1+1}^{k'_1} (\tilde{\Delta}^*_{i,k} + Q_{i,k})$. By summing up both sides of (2.13) over all update steps $k \in (k_1, k'_1]$ yields

$$\tilde{H}_i(\tilde{s}_{i,k'_1+1}) \geq \tilde{H}_i(\tilde{s}_{i,k_1+1}) + \sum_{k=k_1+1}^{k'_1} (\tilde{\Delta}^*_{i,k} + Q_{i,k}). \tag{A.1}$$

Similarly, using Assumption 2.4 and summing both sides of (2.22) over all $k \in (k_1, k'_1 - 1]$ and using (2.21) for $k = k'_1$ yields

$$0 \leq \sum_{k=k_1+1}^{k'_1} \Psi_{i,k} \|d_{i,k}\|^2 \leq \sum_{k=k_1+1}^{k'_1} (\tilde{\Delta}^*_{i,k} + Q_{i,k}). \tag{A.2}$$

By Assumption 2.3, the length of the chosen interval $(k_1, k'_1]$ is always finite. Therefore, any $\{1, \ldots, k_2\}$ with $k_2 < \infty$ can be decomposed into a sequence of similar sub-intervals: $\{(k_{11}, k'_{11}], (k_{12}, k'_{12}], \ldots, (k_{1L}, k'_{1L}]\}$ where $k_{11} = 0$, $k'_{1i} = k_{1(i+1)}$ $\forall i \in (0, L]$. If $k_2$ is such that $k'_{1L} < k_2$ (which happens if $0 > \sum_{k=k'_{1L}+1}^{k_2} Q_{i,k}$), Assumption 2.3 implies that there exists some $k'_2$ such that $k_2 < k'_2 < \infty$ which satisfies $0 < \sum_{k=k'_{1L}+1}^{k'_2} Q_{i,k}$ (i.e., $(k'_{1L}, k'_2]$ is the new last sub-interval of $(0, k'_2]$). Then, by writing the respective expressions in (A.1) and (A.2) for each such sub-interval of the complete interval $(0, k_2']$ and summing both sides over all $k$ yields

$$\tilde{H}_i(\tilde{s}_{i,k_2'+1}) \geq \tilde{H}_i(\tilde{s}_{i,1}) + \sum_{k=1}^{k_2'} (\tilde{\Delta}_{i,k}^* + Q_{i,k}), \tag{A.3}$$

$$0 \leq \sum_{k=1}^{k_2'} \Psi_{i,k} \|d_{i,k}\|^2 \leq \sum_{k=1}^{k_2'} (\tilde{\Delta}_{i,k}^* + Q_{i,k}), \tag{A.4}$$

respectively. Using Assumption 2.1 in (A.3) gives $|\bar{B}_i| H_{UB} \geq \tilde{H}_i(\tilde{s}_{i,k_2'+1}) - \tilde{H}_i(\tilde{s}_{i,1}) \geq \sum_{k=1}^{k_2'} (\tilde{\Delta}_{i,k}^* + Q_{i,k})$. Combining this with (A.4) yields

$$\sum_{k=1}^{k_2'} \Psi_{i,k} \|d_{i,k}\|^2 \leq |\bar{B}_i| H_{UB}. \tag{A.5}$$

By Assumption 2.1, the term $|\bar{B}_i| H_{UB}$ in (A.5) is a finite positive number. Also, by Assumption 2.4, $\Psi_{i,k} \geq \epsilon > 0,\ \forall k$. Therefore, taking limits of the above expression as $k_2' \to \infty$ implies the convergence criterion in (2.6) as long as the optimal step sizes given by (2.17) are used.

### A.2.4 Proof of Theorem 2.2

The proof uses the same steps as in that of Theorem 2.1. The only difference lies in the use of new terms for $\tilde{\Delta}_{i,k}$, $\tilde{\Delta}_{i,k}^*$ and $Q_{i,k}$, given by (2.26), (2.28) and (2.27). Then, the final step of the proof is

$$\sum_{k=1}^{k_2'} \Psi_{i,k} [\mathbf{1}\{i \in \mathcal{N}\} \|d_{i,k}\|^2 + \mathbf{1}\{i \in \mathcal{B}\} \|\hat{d}_{i,k}\|^2] \leq |\bar{B}_i| H_{UB}. \tag{A.6}$$

By Assumption 2.1, the R.H.S. of the above expression is finite and positive. Taking limits when $k_2' \to \infty$ yields the convergence criteria given in (2.7) and (2.8). Further, noting that Theorem 2.2 is a generalization of Theorem 2.1 with the step size selection scheme (2.28) replacing (2.17), it follows that (2.6) is also satisfied.

### A.2.5 Proof of Lemma 2.6

Consider the problem where the neighborhood objective function $\tilde{H}_i(\tilde{s}_{i,k})$ is maximized using the projected state updates of $s_{i,k}$ on the convex feasible space $\mathbf{F}$. Following (Bertsekas, 2016) in this situation, the convergence condition on the step sizes $\beta_{i,k}$ is $0 < \beta_{i,k} < \frac{2}{K_i}$, where $K_i$ is the Lipschitz constant of $\nabla\tilde{H}_i$. Note that $K_i = \sum_{j\in\bar{B}_i} K_{1j}$ due to (2.12). Also, for $i \in \mathcal{N}$, the expression for $\beta^*_{i,k}$ given in (2.28) can be modified into the form

$$\beta^*_{i,k} = \frac{1}{K_i}\left[1 + \frac{d^T_{i,k}\sum_{j\in B_i} d_{ij,k}}{\|d_{i,k}\|^2}\right]. \tag{A.7}$$

Now, enforcing the convergence condition $0 < \beta^*_{i,k} < \frac{2}{K_i}$ yields the first condition in (2.30). Similarly the second condition in (2.30) can be obtained when the expression for $\beta^*_{i,k}$, $i \in \mathcal{B}$, in (2.28) is considered.

### A.2.6 Proof of Lemma 2.7

By taking the partial derivative of (2.40) (written for agent $j$) w.r.t. the local state $s_i$ yields

$$d_{ij} = -\int\limits_{V_j} R(x)p_j(x,s_j)\prod_{l\in B_j-\{i\}}(1-p_l(x,s_l))\frac{dp_i(x,s_i)}{ds_i}dx.$$

Now, note that $\forall x \notin V_i,\ \frac{-dp_i(x,s_i)}{ds_i} = 0$, and, $\forall l \notin B_i \cap B_j, \forall x \in V_i \cap V_j,\ p_l(x,s_l) = 0$. By incorporating these relationships into the obtained expression for $d_{ij}$ gives a locally computable (at agent $i$) expression for $d_{ij}$ as

$$d_{ij} = -\int\limits_{V_i\cap V_j} R(x)p_j(x,s_j)\prod_{l\in B_i\cap B_j}(1-p_l(x,s_l))\frac{dp_i(x,s_i)}{ds_i}dx.$$

## A.3 From Chapter 3

### A.3.1 Proof of Lemma 3.1

Take the optimal solution of (3.1) as $Y^* = \{y_1, y_2, y_3, \ldots, y_N\}$. Due to the monotonicity of $f$, $f(Y^*) \leq f(Y^* \cup Y^G)$. Note also that $\Delta f(Y^*|Y^G) = f(Y^* \cup Y^G) - f(Y^G)$. Therefore,

$$f(Y^*) \leq f(Y^G) + \Delta f(Y^*|Y^G). \tag{A.8}$$

Now, consider the $\Delta f(Y^*|Y^G)$ term which can be written as a telescopic sum:

$$\begin{aligned}\Delta f(Y^*|Y^G) = {} & [f(\{y_1\} \cup Y^G) - f(Y^G)] + [f(\{y_1, y_2\} \cup Y^G) - f(Y^G \cup \{y_1\})] + \ldots \\ & + [f(\{y_1, \ldots, y_N\} \cup Y^G) - f(Y^G \cup \{y_1, \ldots, y_{N-1}\})].\end{aligned}$$

Using the marginal gain function notation (see Def. 3.1), this can be written as,

$$\Delta f(Y^*|Y^G) = \sum_{i=1}^{N} \Delta f(y_i|Y^G \cup \{y_1, y_2, \ldots, y_{i-1}\}). \tag{A.9}$$

Next, using the formed set $E_1$ given in (3.11), $\Delta f(y_1|Y^G)$ term can be upper bounded with the largest value in $E_1$ (i.e., $\alpha^1_{d1}$), as

$$\Delta f(y_1|Y^G) \leq \alpha^1_{d1}. \tag{A.10}$$

As a result of the submodularity property of $f$, $\Delta f(y_2|Y^G \cup \{y_1\}) \leq \Delta f(y_2|Y^G)$. Since $y_1 \neq y_2$ and (A.10), $\Delta f(y_2|Y^G)$ can be upper bounded using $\alpha^2_{d1}$. Therefore,

$$\Delta f(y_2|Y^G \cup \{y_1\}) \leq \Delta f(y_2|Y^G) \leq \alpha^2_{d1}.$$

Similarly, for all $i \in \{1, 2, \ldots N\}$, $\Delta f(y_i | Y^G \cup \{y_1, y_2, \ldots, y_{i-1}\}) \leq \alpha_{d1}^i$. Applying this result in (A.9) and using (3.12) gives

$$\Delta f(Y^* | Y^G) \leq \sum_{j=1}^{N} \alpha_{d1}^j = \alpha_{d1}. \tag{A.11}$$

Finally, using (A.11) in (A.8) yields $f(Y^*) \leq f(Y^G) + \alpha_{d1}$. Therefore,

$$\frac{f(Y^*)}{f(Y^G)} \leq 1 + \frac{\alpha_{d1}}{f(Y^G)} \iff \left[1 + \frac{\alpha_{d1}}{f(Y^G)}\right]^{-1} \leq \frac{f(Y^G)}{f(Y^*)}.$$

### A.3.2 Proof of Lemma 3.2

Note that the set function $f$ is of the form $f : 2^X \to \mathbb{R}$ while the set function $\Psi$ is of the form $\Psi : 2^{X \backslash F^G} \to \mathbb{R}$. Since $Y^G$ is a known fixed set, $f(Y^G)$ is a known constant. Therefore, $\Psi(Y)$ can be thought of as a set function that follows $f(Y^G \cup Y)$ while having a constant offset of $f(Y^G)$ for all $Y \in 2^{X \backslash F^G}$. Therefore, $\Psi$ inherits the submodularity and monotonicity properties from $f$. Moreover, from the monotonicity property of $f$, $\Psi(Y) \geq 0$ and $\Psi(Y) = 0$ occurs only when $Y = \emptyset$. Therefore $\Psi$ is normalized. Hence $\Psi(Y)$ is a polymatroid function.

The set-function constraint in the problem (3.15) can be seen as a matroid ($\mathcal{M}_\# = (X \backslash Y^G, \mathcal{I}_\#)$) constraint. In fact, here, $\mathcal{M}_\#$ is a uniform matroid of rank $N$. Therefore, Theorem 3.1 is applicable to (3.15) and thus the performance bound in (3.16) fallows.

### A.3.3 Proof of Lemma 3.3

Consider the inequality in (A.8): $f(Y^*) \leq f(Y^G) + \Delta f(Y^* | Y^G)$. The motivation behind this lemma is to provide an alternative upper bound to the $\Delta f(Y^* | Y^G)$ term (different from (A.11)) using the information obtained from running $N$ additional greedy iterations (i.e. $2N$ greedy iterations in total, recall that $|X| = n \geq 2N + 1$).

Using the set function $\Psi(Y)$ in (3.15), note that $\Delta f(Y^*|Y^G)$ can be written as

$$\Delta f(Y^*|Y^G) = f(Y^G \cup Y^*) - f(Y^G) = \Psi(Y^*). \tag{A.12}$$

Note also that (3.16) in Lemma 3.2 gives

$$\Psi(Y^*_\#) \leq \frac{1}{\beta_{f\#}} \Psi(Y^G_\#). \tag{A.13}$$

Since $\Psi(Y^*_\#)$ is the global maximum of the problem (3.15), $\Psi(Y^*) \leq \Psi(Y^*_\#)$.

Moreover, the greedy solution to (3.15) is $Y^G_\# = Y^{G2} \backslash Y^G$ where $Y^{G2}$ is the greedy solution to (3.1) when $2N$ greedy iterations are executed. Therefore, using the definition of $\Psi$ in (3.15), $\Psi(Y^G_\#) = f(Y^{G2}) - f(Y^G)$ is obtained.

Now, using these relationships, (A.13) can be developed into

$$\Psi(Y^*) \leq \Psi(Y^*_\#) \leq \frac{1}{\beta_{f\#}} \Psi(Y^G_\#) = \frac{1}{\beta_{f\#}} (f(Y^{G2}) - f(Y^G)).$$

This result can be re-stated using (3.14) and (A.12) as

$$\Delta f(Y^*|Y^G) \leq \alpha_{d2}. \tag{A.14}$$

The proof is complete by noticing the similarity between (A.14) and (A.11) (also between (3.17) and (3.13)) and thus following the last step in proof of Lemma 3.1.

### A.3.4 Proof of Theorem 3.4

Since $f(Y^G) \geq 0$, the proof will be complete if $f(Y^*) \leq \alpha_m$ is established. Take the optimal solution of (3.1) as $Y^* = \{y_1, y_2, \ldots, y_N\}$. Note that $f(Y^*)$ can be written as a telescopic sum:

$$\begin{aligned} f(Y^*) &= f(\{y_1, y_2, \ldots, y_N\}), \\ &= \Delta f(y_N|\{y_1, y_2, \ldots, y_{N-1}\}) + f(\{y_1, y_2, \ldots, y_{N-1}\}), \\ &\vdots \\ f(Y^*) &= \sum_{i=1}^{N} \Delta f(y_i|\{y_1, y_2, \ldots, y_{i-1}\}). \end{aligned} \tag{A.15}$$

Now, the submodularity property of $f$ can be used to write

$$\Delta f(y_i|y_1, y_2, \ldots, y_{i-1}) \leq \Delta f(y_i|\emptyset), \quad \forall i \in \{1, 2, \ldots, N\}. \tag{A.16}$$

Therefore, (A.15) can be upper bounded using (A.16) as,

$$f(Y^*) \leq \sum_{i=1}^{N} \Delta f(y_i|\emptyset) \leq \max_{\substack{B:B\subseteq X \\ |B|=N}} \left[ \sum_{x_i \in B} \Delta f(x_i|\emptyset) \right] = \alpha_m \tag{A.17}$$

(the last step is a result of the definition of $\alpha_m$ in (3.22)). This completes the proof.

### A.3.5 Proof of Corollary 3.1

Since the set-coverage function $H(\cdot)$ is submodular, the first statement in Def. 3.2:

$$H(A \cup B) + H(A \cap B) \leq H(A) + H(B), \tag{A.18}$$

applies. Using this and the simple fact $H(A \cap B) \geq 0$ completes the first part of the proof. Next, replacing $A$ and $B$ in (A.18) respectively with $A \cup B$ and $A \cup C$ gives

$$\begin{aligned} H(A \cup B) + H(A \cup C) &\geq H((A \cup B) \cup (A \cup C)) + H((A \cup B) \cap (A \cup C)) \\ &= H(A \cup B \cup C) + H(A \cup (B \cap C)) \\ &\geq H(A \cup B \cup C) + H(A), \end{aligned}$$

where $H(A \cup (B \cap C)) \geq H(A)$ (from monotonicity) has been used in the final step.

### A.3.6 Proof of Theorem 3.6

First, note that $H(A \cup \{s_i\})$ can be evaluated using (3.27) as

$$\begin{aligned} H(A \cup \{s_i\}) &= \int_F R(x)(1 - \prod_{s_j \in (A \cup \{s_i\})} (1 - p_j(x, s_j)))dx \\ &= \int_F R(x)(1 - (1 - p_i(x, s_i)) \prod_{s_j \in A} (1 - p_j(x, s_j)))dx \\ &= \underbrace{\int_F R(x)(1 - \prod_{s_j \in A} (1 - p_j(x, s_j)))dx}_{= \, H(A)} \\ &\quad + \underbrace{\int_F R(x)p_i(x, s_i) \prod_{s_j \in A} (1 - p_j(x, s_j)))dx}_{=H(s_i|A)} \end{aligned}$$

Therefore,

$$\Delta H(s_i|A) = H(A \cup \{s_i\}) - H(A) = \int_F R(x)p_i(x, s_i) \prod_{s_j \in A} (1 - p_j(x, s_j)))dx.$$

Now, note that $\forall x \ni p_i(x, s_i) > 0$, the term $(1 - p_j(x, s_j)) \neq 1$ only when $p_i(x, s_i)p_j(x, s_j) > 0$ (i.e., when $j$ is in the neighbor set $B_i$ within the agent set $A$). Therefore,

$$\Delta H(s_i|A) = \int_F R(x)p_i(x, s_i) \prod_{j \in B_i} (1 - p_j(x, s_j)))dx$$

and thus, $\Delta H(s_i|A)$ depends only on the neighborhood state $\bar{s}_i$.

### A.3.7 Proof of Theorem 3.7

For some fixed $s_i \in X$, consider two sets $A, B$ such that $B \subseteq A \subseteq (X \backslash \{s_i\})$ and an element $s_k \in (X \backslash (A \cup \{s_i\}))$ that defines $\Phi^i_{k|A} \triangleq \Delta H(s_i|A \cup \{s_k\}) - \Delta H(s_i|A)$, where

$$\begin{aligned}\Phi^i_{k|A} =& \int\limits_F R(x)p_i(x,s_i) \prod_{s_j \in A \cup \{s_k\}} (1 - p_j(x,s_j)))dx \\ &- \int\limits_F R(x)p_i(x,s_i) \prod_{s_j \in A} (1 - p_j(x,s_j)))dx \\ =& - \int\limits_F R(x)p_i(x,s_i)p_k(x,s_k) \prod_{s_j \in A} (1 - p_j(x,s_j)))dx.\end{aligned}$$

Similarly, defining $\Phi^i_{k|B} \triangleq \Delta H(s_i|B \cup \{s_k\}) - \Delta H(s_i|B)$, it can be shown that

$$\Phi^i_{k|B} = - \int\limits_F R(x)p_i(x,s_i)p_k(x,s_k) \prod_{s_j \in B} (1 - p_j(x,s_j)))dx.$$

Note also that $\forall x \in F$ whenever $B \subseteq A$,

$$\prod_{s_j \in A} (1 - p_j(x,s_j)) \leq \prod_{s_j \in B} (1 - p_j(x,s_j)). \tag{A.19}$$

The above three results can be used to conclude $\Phi^i_{k|B} \leq \Phi^i_{k|A}$, i.e.,

$$\Delta H(s_i|B \cup \{s_k\}) - \Delta H(s_i|B) \leq \Delta H(s_i|A \cup \{s_k\}) - \Delta H(s_i|A).$$

This implies that $-\Delta H(s_i|A)$ is submodular (i.e., $\Delta H(s_i|A)$ is supermodular) in $A$.

Now, to establish the monotonicity related property, consider

$$\begin{aligned}\Delta H(s_i|A) - \Delta H(s_i|B) =& \int\limits_F R(x)p_i(x,s_i) \prod_{\forall s_j \in A} (1 - p_j(x,s_j)))dx \\ &- \int\limits_F R(x)p_i(x,s_i) \prod_{\forall s_j \in B} (1 - p_j(x,s_j)))dx\end{aligned}$$

Using (A.19), it is clear that $\Delta H(s_i|A) - \Delta H(s_i|B) \leq 0$. Therefore, $-\Delta H(s_i|A)$ is monotone (i.e., $\Delta H(s_i|A)$ is non-increasing) in $A \subseteq (X\backslash\{s_i\})$ for some fixed $s_i \in X$.

### A.3.8 Proof of Lemma C.1

Negating and adding a constant $H(A)$ term to the definition given in (C.1) results in

$$A^- = \underset{\substack{B:B\subset A,\\|B|=|A|-1}}{\operatorname{argmin}} [H(A) - H(B)].$$

Now, the variable $B$ in the above optimization problem is changed by taking $B = A\backslash\{a\}$ and considering $a \in A$ as the new variable. Thus, $A^- = A\backslash\{a^*\}$ with

$$a^* = \underset{a\in A}{\operatorname{argmin}} \; [H(A) - H(A\backslash\{a\})] = \underset{a\in A}{\operatorname{argmin}} \; \Delta H(a|A\backslash\{a\}).$$

### A.3.9 Proof of Lemma C.2

Consider the R.H.S. of the given statement (also, take $s_{Aw} = A\backslash A^-$),

$$\begin{aligned}
R.H.S. &= H(B) - H(B\backslash\{s_j\}) = \; \Delta H(s_j|B\backslash\{s_j\})\\
&\geq \Delta H(s_j|A\backslash\{s_j\}), \text{(using the monotonicity of } -\Delta(s_j|\cdot\backslash\{s_j\}))\\
&\geq \Delta H(s_j|A\backslash\{s_j\}) - \Delta H(s_{Aw}|A\backslash s_{Aw}), \; \text{(Since } H(s_{Aw}|A\backslash s_{Aw}) \geq 0)\\
&= H(A) - \Delta H(s_{Aw}|A\backslash s_{Aw}) - H(A) + \Delta H(s_j|A\backslash\{s_j\}), \; \text{(Add/remove } H(A))\\
&= H(A\backslash s_{Aw}) - H(A\backslash\{s_j\})\\
&= H(A^-) - H(A\backslash\{s_j\})\\
&= L.H.S.
\end{aligned}$$

### A.3.10 Proof of Lemma C.3

Consider the L.H.S. of the given expression that can be simplified using (C.1) as

$$H(A) - H(A^-) = H(A) - \max_{\substack{C:C\subset A,\\|C|=|A|-1}} [H(C)] = \max_{\substack{C:C\subset A,\\|C|=|A|-1}} [H(A) - H(C)]\,.$$

Changing the variable of this optimization problem from $C$ to $s_j$ using: $C = A\backslash\{s_j\}$,

$$H(A) - H(A^-) = \max_{s_j \in A} [H(A) - H(A\backslash\{s_j\})] = \max_{s_j \in A} \ \Delta H(s_j|A\backslash\{s_j\}).$$

Since $A^- \neq B$ is given, the optimal $s_j$ is in the set $B$. Therefore,

$$H(A) - H(A^-) = \max_{s_j \in B} \ \Delta H(s_j|A\backslash\{s_j\}). \tag{A.20}$$

Similarly, the R.H.S. of the given expression can be transformed into the form:

$$H(B) - H(B^-) = \max_{s_j \in B} \ \Delta H(s_j|B\backslash\{s_j\}). \tag{A.21}$$

Now, from the monotonicity property of $-\Delta H(s_j|B\backslash\{s_j\})$ w.r.t. $B$,

$$\Delta H(s_j|A\backslash\{s_j\}) \leq \Delta H(s_j|B\backslash\{s_j\}), \quad \forall s_j \in B.$$

Therefore,

$$\max_{s_j \in B} \ \Delta H(s_j|A\backslash\{s_j\}) \leq \max_{s_j \in B} \ \Delta H(s_j|B\backslash\{s_j\}). \tag{A.22}$$

Finally, the above three key results (i.e., (A.20), (A.21) and (A.22)) can be used to obtain the required relationship: $H(A) - H(A^-) \leq H(B) - H(B^-)$.

### A.3.11 Proof of Lemma C.4

Take the worst contributors of $A_{k+1}$ and $A_k$ as $\omega_{k+1}$ and $\omega_k$, respectively. Therefore,

$$A_{k+1} = \{\omega_{k+1}\} \cup A_k \quad \text{and} \quad A_k = \{w_k\} \cup A_{k-1}. \tag{A.23}$$

Since $A_{k+1}^- = A_k$, $\Delta H(\omega_{k+1}|A_k) \leq \Delta H(\omega|A_{k+1}\backslash\omega)$, $\forall \omega \in A_{k+1}$. Therefore, clearly, $\Delta H(\omega_{k+1}|A_k) \leq \Delta H(\omega|A_{k+1}\backslash\omega)$, $\forall \omega \in A_k$. Here, consider the case when $\omega = \omega_k$: $\Delta H(\omega_{k+1}|A_k) \leq \Delta H(\omega_k|A_{k+1}\backslash\omega_k)$. Now, using the definition of marginal-coverage gain $\Delta H(\cdot|\cdot)$, the relationship: $H(A_{k+1}) - H(A_k) \leq H(A_{k+1}) - H(A_{k+1} \backslash \omega_k)$ can be obtained that implies: $H(A_{k+1} \backslash \omega_k) \leq H(A_k)$. Next, since $A_{k+1} \backslash \omega_k = \{\omega_{k+1}\} \cup A_{k-1}$,

$$H(\{\omega_{k+1}\} \cup A_{k-1}) \leq H(A_k). \tag{A.24}$$

Note that (A.23) implies $A_{k+1} = \{\omega_{k+1}, \omega_k\} \cup A_{k-1}$. Therefore,

$$\begin{aligned} H(A_{k+1}) + H(A_{k-1}) &= H(\{\omega_{k+1}, \omega_k\} \cup A_{k-1}) + H(A_{k-1}), \\ &\leq H(\{\omega_{k+1}\} \cup A_{k-1}) + H(\{\omega_k\} \cup A_{k-1}), \text{(using Corollary 3.1)} \\ &\leq H(A_k) + H(A_k), \text{(using (A.24) and (A.23))}. \end{aligned}$$

Thus, $H(A_{k+1}) - H(A_k) \leq H(A_k) - H(A_{k-1})$.

### A.3.12 Proof of Theorem C.1

Notice that compared to lemma C.3, both conditions $A^- \neq B$ and $|B| = |A| - 1$ are omitted in this theorem.

Therefore, first, the condition $|B| = |A| - 1$ is assumed. Then, it needs to be shown that the given relationship holds when $A^- = B$. For this purpose, consider the three distinct sets $A, A^- = B, B^-$, which in this scenario, are respectively equivalent to the three sets $A_{k+1}, A_k, A_{k-1}$ discussed in Lemma C.4. Therefore, Lemma C.4 yields

$$H(A_{k+1}) - H(A_k) \leq H(A_k) - H(A_{k-1}) \implies H(A) - H(A^-) \leq H(B) - H(B^-). \tag{A.25}$$

Second, the assumption $|B| = |A| - 1$ is relaxed, and thus, $|B|$ is now allowed to be smaller than $|A| - 1$ (as still $B \subset A$). However, in a such situation, note that the relationship in (A.25) can be extended (i.e., iteratively applied) to prove

$$H(A) - H(A^-) \leq H(A_1) - H(A_1^-) \leq H(A_2) - H(A_2^-) \leq \cdots \leq H(B) - H(B^-),$$

where $B \subset \cdots \subset A_2 \subset A_1 \subset A$ and the difference in the cardinality of any two consecutive sets is always 1. Therefore, irrespective of the condition $|B| = |A| - 1$,

$$H(A) - H(A^-) \leq H(B) - H(B^-).$$

## A.4 From Chapter 4

### A.4.1 Proof of Lemma 4.2

At $k = k_{eq}$, in (4.10), $\bar{\tau}_{k+1} = \bar{\tau}_k = \bar{\tau}_{eq}$. Therefore, $\bar{\tau}_{eq} = (\Delta_1 - \Delta_2)^{-1}\bar{1}_m\rho_\Xi$. Using $\Delta_1 = diag(\bar{\alpha}) - \Delta_2^T$ and $diag(\bar{1}_m) + \Delta_2^T + \Delta_2 = \bar{1}_m\bar{1}_m^T$, $\bar{\tau}_{eq}$ can be simplified as:

$$\bar{\tau}_{eq} = (diag(\bar{\alpha} + \bar{1}_m) - \bar{1}_m\bar{1}_m^T)^{-1}\bar{1}_m\rho_\Xi. \tag{A.26}$$

Note that $\alpha_n$ and $\beta_n$ satisfy $(\alpha_n + 1)^{-1} = \beta_n \iff (diag(\bar{\alpha} + \bar{1}_m))^{-1} = diag(\bar{\beta})$. Also, $diag(\bar{\beta})\bar{1}_m = \bar{\beta}$ and $I_m \in \mathbb{R}^{m\times m}$ is an identity matrix. Now, using Lemma 4.1,

$$\bar{\tau}_{eq} = diag(\bar{\beta})\left(I_m + \frac{\bar{1}_m\bar{1}_m^T diag(\bar{\beta})}{1 - \bar{1}_m^T\bar{\beta}}\right)\bar{1}_m\rho_\Xi = \left(\frac{\bar{\beta}}{1 - \bar{1}_m^T\bar{\beta}}\right)\rho_\Xi.$$

Components of $\bar{\tau}_{eq}$ are non-negative only when $1 - \bar{1}_m^T\bar{\beta} > 0$. Thus, using the definition of $\bar{\beta}$, the condition $\bar{1}_m^T\bar{\beta} = \sum_{i=1}^m \frac{A_i}{B_i} < 1$ can be obtained.

### A.4.2 Proof of Lemma 4.3

Let $\bar{e}_k = \bar{\tau}_k - \bar{\tau}_{eq}$ be the steady state error. Then, $\bar{e}_{k+1} = \bar{\tau}_{k+1} - \bar{\tau}_{eq}$. Now, using (4.10) and Lemma 4.2, $\bar{e}_{k+1} = (\Delta_1^{-1}\Delta_2\bar{\tau}_k + \Delta_1^{-1}\bar{1}_m\rho_\Xi) - (\Delta_1^{-1}\Delta_2\bar{\tau}_{eq} + \Delta_1^{-1}\bar{1}_m\rho_\Xi)$, so that, $\bar{e}_{k+1} = \Delta_1^{-1}\Delta_2\bar{e}_k$. Therefore, under Assumption 4.3 the equilibrium point $\bar{\tau}_{eq}$ given in (4.11) of (4.10) is globally asymptotically stable (Bof et al., 2018).

### A.4.3 Proof of Theorem 4.1

Under the given conditions, Lemmas 4.2 and 4.3 are applicable. Thus, (4.7) gives

$$J_{ss}(\Xi) = \lim_{T\to\infty}\frac{1}{T}\int_0^T \sum_{n=1}^m R_n(t)dt = \frac{1}{T_\Xi}\int_{\partial T_\Xi}\sum_{n=1}^m R_n(t)dt = \sum_{n=1}^m \frac{1}{T_\Xi}\int_{\partial T_\Xi} R_n(t)dt,$$

where $T_\Xi \triangleq \rho_\Xi + \bar{1}_m^T\bar{\tau}_{eq}$ represents the steady state tour duration and $\partial T_\Xi$ is a time period of a tour occurring after achieving steady state. Using the $R_n(t)$ trajectory shown in Fig. 4·4 note that when equilibrium is achieved (as $T, k \to \infty$), the final tour uncertainties will become stationary (i.e., $R_{n,k} = R_{n,k+1}, \forall n \in \Xi$). Hence the area under the $R_n(t)$ trajectory evaluated over a period $T_\Xi$ becomes equivalent to that of a triangle where the base is $T_\Xi$ and the height is $(B_n - A_n)\tau_{n,\infty}, \forall n \in \Xi$. Therefore,

$$J_{ss}(\Xi) = \sum_{n=1}^{m} \frac{1}{T_\Xi}\frac{1}{2}T_\Xi(B_n - A_n)\tau_{n,\infty} = \frac{1}{2}(\bar{B} - \bar{A})^T\bar{\tau}_{eq}.$$

### A.4.4 Proof of Lemma 4.4

The mean system uncertainty $J_T$ in (4.2) (for the original PMN problem setting with $M$ targets in $\mathcal{V} = \{1, 2, \dots, M\}$) can be decomposed as $J_T = \frac{1}{T}\int_0^T \sum_{j \in \mathcal{V}\setminus\{i\}} R_j(t)dt + \frac{1}{T}\int_0^T R_i(t)dt$, where the second term represents the contribution of target $i$ to the main objective $J_T$. Since target $i$ is not being visited by any agent during $t \in [0, T]$ and from (4.1), $\dot{R}_i(t) = A_i,\ \forall t \in [0, T]$. Also note that the initial target uncertainty of target $i$ is $R_i(0) = R_{i,0}$. Therefore, the contribution of target $i$ can be simplified as

$$\frac{1}{T}\int_0^T R_i(t)dt = \frac{1}{T}\int_0^T (R_{i,0} + A_i t)dt = \left(R_{i,0} + \frac{A_i T}{2}\right).$$

### A.4.5 Proof of Theorem 4.2

When target $i$ is neglected, Lemma 4.4 gives the mean system uncertainty as $(R_{i,0} + \frac{A_i T}{2}) + J_T(\Xi)$. After the target-cycle expansion, the mean system uncertainty is $J_T(\Xi')$ (Note that $i \in \Xi'$). Therefore, the gain in mean system uncertainty is $\left(R_{i,0} + \frac{A_i T}{2}\right) + J_T(\Xi) - J_T(\Xi')$. Now, adding and subtracting a $(J_{ss}(\Xi) - J_{ss}(\Xi'))$ term and applying Assumption 4.4 twice (for $J_T(\Xi)$, $J_T(\Xi')$ terms) shows that the above "gain" can be estimated by the marginal gain expression given in (4.15) (with a tolerance of $\pm 2K_e$).

### A.4.6 Proof of Lemma 4.5

By inspection of the $R_n(t)$ profile in Fig. 4·11, for each target $n \in \bar{\Xi}$ and for each auxiliary target $n^j \in \mathcal{T}_n$, considering its corresponding sub-cycle $\Xi_n^j$'s time period: $(B_n - A_n)\tau_n^j = A_n(T_n^j - \tau_n^j) \iff B_n\tau_n^j = A_n T_n^j$, where $T_n^j$ is the total time taken to complete the sub-cycle $\Xi_n^j$. Now, using the sub-cycle unit vectors, $T_n^j$ can be replaced to get: $B_n\tau_n^j = A_n(\bar{1}_n^j)^T(\bar{\rho}_\Xi + \bar{\tau}_\Xi)$. This relationship gives $|\Xi|$ equations which need to be solved for $\bar{\tau}_\Xi \in \mathbb{R}^{|\Xi|}$. Arranging all the equations in a matrix form: $diag(\bar{\gamma}_\Xi)\bar{\tau}_\Xi = \mathbf{1}_\Xi(\bar{\rho}_\Xi + \bar{\tau}_\Xi)$ gives the result in (4.16).

### A.4.7 Proof of Lemma 4.6

Observing the auxiliary target uncertainty profiles $R_n^1(t)$ and $R_n^2(t)$ (of $n^1$ and $n^2$) in Fig. 4·11 of the target-cycle shown in Fig. 4·10, note that the shape of these profiles should satisfy the previously established equivalence criteria. Therefore, for any generic target-cycle $\bar{\Xi}$, the first equivalence criterion is guaranteed by:

$$B_n^j\tau_n^j = A_n^j T_n^j, \quad \forall n^j \in \mathcal{T}_n, \ \forall n \in \bar{\Xi}. \tag{A.27}$$

Using (4.2), the contribution from a target $n \in \bar{\Xi}$ to the main objective $J_T$ during a tour (over a period of length $T_\Xi$, at steady state) can be written as $\frac{1}{T}\int_{T_\Xi} R_n(t)dt$. Therefore, the condition to enforce the third equivalence criterion is: $\frac{1}{T}\int_{T_\Xi} R_n(t)dt = \frac{1}{T}\int_{T_\Xi}\sum_{n^j\in\mathcal{T}_n} R_n^j(t)dt$. However, since $T_\Xi$ can be decomposed into sub-cycle time periods: $\frac{1}{T}\int_{T_\Xi} R_n(t)dt = \frac{1}{T}\sum_{n^j\in\mathcal{T}_n}\int_{T_n^j} R_n(t)dt$. These two relationships give a system of equations: $\int_{T_\Xi} R_n^j(t)dt = \int_{T_n^j} R_n(t)dt,\ \forall n^j \in \mathcal{T}_n,\ \forall n \in \bar{\Xi}$. As uncertainty profiles are piece-wise linear, these integrals can be evaluated leading to the system of equations:

$$T_\Xi(B_n^j - A_n^j) = T_n^j(B_n - A_n), \quad \forall n^j \in \mathcal{T}_n, \ \forall n \in \bar{\Xi}. \tag{A.28}$$

Finally, (A.27) and (A.28) can be solved to obtain the auxiliary target parameters.

## A.5 From Chapter 5

### A.5.1 Proof of Theorem 5.1

In (5.3), by taking the summation operator out of the integration and then splitting the time interval $[0, T]$ into three parts gives

$$J_T = \frac{1}{T}\left[\sum_{j\in\mathcal{T}\backslash\{i\}} \int_0^T R_j(t)dt\right] + \frac{1}{T}\left[\int_0^{t_0} R_i(t)dt + \int_{t_0}^{t_1} R_i(t)dt + \int_{t_1}^{T} R_i(t)dt\right].$$

Moreover, since $[t_0, t_1) \subseteq [t_i^k, t_i^{k+1})$, the relationship (5.5) implies that $\int_{t_0}^{t_1} R_i(t)dt$ represents the area of a trapezoid (whose parallel sides are $R_i(t_0)$ and $R_i(t_1)$). Therefore, $J_i(t_0, t_1) = \frac{R_i(t_0)+R_i(t_1)}{2}(t_1 - t_0)$. Also, it follows from (5.1) and (5.5) that $R_i(t_1) = R_i(t_0) + \dot{R}_i(t_0)(t_1 - t_0)$. Combining these two results gives (5.7).

### A.5.2 Proof of Lemma 5.1

Using (5.21), first and second order derivatives $J'(u_j)$ and $J''(u_j)$ of $J_H(u_j, 0)$ can be obtained respectively as

$$J'(u_j) = \frac{\bar{A} - B_j}{2} + \frac{B_j\rho_{ij}^2}{2(\rho_{ij} + u_j)^2} \quad \text{and} \quad J''(u_j) = -\frac{B_j\rho_{ij}^2}{(\rho_{ij} + u_j)^3}.$$

Observe that $J'(0) = \bar{A}/2 > 0$ and $J''(u_j) < 0$, $\forall u_j \geq 0$. This implies that $J'(u_j)$ is monotonically decreasing with $u_j \geq 0$. Also note that $\lim_{u_j\to\infty} J'(u_j) = \frac{\bar{A}-B_j}{2}$. Therefore, for the case where $\bar{A} \geq B_j$, the objective $J_H(u_j, 0)$ is monotonically increasing with $u_j$. Hence $u_j^* = 0$ in (5.24).

For the case where $\bar{A} < B_j$, the limiting value of $J'(u_j)$ is negative. This implies the existence of a maximum of $J_H(u_j, 0)$ at some $u_j \geq 0$. However, such a maximizing $u_j$ value is irrelevant to the minimization in (5.24). The existence of this maximum and $J''(u_j) < 0$ imply the existence of a point $u_j = u_j^{\#}$ such that $J_H(0, 0) = J_H(u_j, 0)$ occurs. Using (5.21), this can be determined as $u_j^{\#} = \frac{C_6 - C_4\rho_{ij}}{C_1}$ which simplifies to $u_j^{\#}$

given in (5.26). According to the nature of $J'(u_j)$ and $J''(u_j)$, it is then clear that $J_H(u_j, 0)$ decreases with $u_j \geq u_j^{\#}$ (below its $J_H(0,0)$ value). Therefore, when $\bar{u}_j \geq u_j^{\#}$ (and $\bar{A} < B_j$), $u_j^* = \bar{u}_j$ in (5.24).

### A.5.3 Proof of Lemma 5.2

The first and second order derivatives of $J_H(u_j^B, v_j)$ with respect to $v_j$ are

$$J'(0) = \frac{\bar{A}}{2} - \frac{B_j}{2}\left[1 - \frac{\rho_{ij}^2}{(\rho_{ij} + u_j^B)^2}\right] \quad \text{and} \quad J''(v_j) \quad = \frac{R_j^2(t) + 2B_j\rho_{ij}R_j(t) + A_jB_j\rho_{ij}^2}{(B_j - A_j)(\rho_{ij} + u_j^B + v_j)},$$

respectively. Note that $J''(v_j) > 0,\ \forall v_j \geq 0$. This implies that $J_H(u_j^B, v_j)$ is convex in the positive orthant of $v_j$, and $J'(v_j)$ is increasing with $v_j \geq 0$ starting from $J'(0)$ given above. If $J'(0) \geq 0$, this implies that $J_H(u_j^B, v_j)$ is monotonically increasing with $v_j \geq 0$. Therefore, in this case $v_j^* = 0$, which proves the first case in (5.29).

When $J'(0) < 0$, there must exist a unique minimum to $J_H(u_j^B, v_j)$ at some $v_j \geq 0$. It is straightforward to determine the minimizing $v_j$ value which is found to be $v_j = v_j^{\#}$ given in (5.30). Based on the constraint $0 \leq v_j \leq \bar{v}_j$ in (5.29) and the convexity of $J_H(u_j^B, v_j)$, it is clear that whenever $v_j^{\#} \leq \bar{v}_j$, $v_j^* = v_j^{\#}$ in (5.29) and whenever $v_j^{\#} > \bar{v}_j$, $v_j^* = \bar{v}_j$. This proves the second case given in (5.29).

### A.5.4 Proof of Lemma D.1

The first and second order derivatives of $h(r)$ are

$$h' = \frac{gf' - fg'}{g^2} \quad \text{and} \quad h'' = \frac{g[gf'' - fg''] - 2g'[gf' - fg']}{g^3}.$$

Note that $h''(r) = \frac{\Delta_h(r)}{g^3(r)}$ and $g^3(r) > 0\ \forall r \in \mathcal{U}$. Therefore, convexity of $h(r)$ will only depend on the condition: $h''(r) > 0,\ \forall r \in \mathcal{U} \iff \Delta_h(r) > 0, \forall r \in \mathcal{U}$. This condition is easily seen to be satisfied whenever

$$\Delta_h(r_0) > 0 \text{ for some } r_0 \in \mathcal{U} \text{ and } \Delta_h'(r) = 0 \text{ for all } r \in \mathcal{U}.$$

Finally, evaluating $\Delta_h'(r)$ yields the expression in (D.1)

$$\Delta_h'(r) = g[gf''' - fg'''] - 3g''[gf' - fg'],$$

which completes the proof.

### A.5.5 Proof of Lemma 5.3

Recall that $U_{ij} = [u_j, v_j]$ and $w = \rho_{ij} + u_j + v_j$ for **RHCP3**. Applying $\alpha = 0$ in (5.49) and using it in the **RHCP3** objective $J_H(U_{ij}) = \frac{1}{w}\bar{J}_i(t, t+w)$ gives $J_H(U_{ij}) = \bar{R}_j(t) + \frac{1}{2}\bar{A}_j(\rho_{ij} + u_j + v_j)$. Therefore, clearly the minimizing $U_{ij}$ choice is $u_j^* = v_j^* = 0$ (i.e., the solution to (5.11)). Hence, the optimal next-visit target $j^*$ following from (5.12) is $j \in \mathcal{N}_i$ with the minimum $\bar{R}_j(t) + \frac{1}{2}\bar{A}_j\rho_{ij}$ value. Using the relationships $\bar{R}_j = \bar{R} - R_j$ and $\bar{A}_j = \bar{A} - A_j$ (see (5.22)), this $j^*$ choice yields (5.50).

# Appendix B

# Concepts Adopted From Literature

## B.2 In Chapter 2

### B.2.1 Boosting Function Families Proposed in (Sun et al., 2014) (used in Section 2.3.5)

**$\Phi$-Boosting:** This method uses $\alpha_{i1}(x, \bar{s}_i) = \kappa \Phi_i(x)^\gamma$ and $\eta_{i1}(x, \bar{s}_i) = 0$, where $\Phi_i(x)$ in (2.42) indicates the extent to which point $x \in V_i$ is *not* covered by neighbors in $B_i$. Thus, the effect of $\Phi$-Boosting is to force agent $i$ to move towards regions of $V_i$ which are less covered by its neighbors in $B_i$.

***P*-Boosting:** In this method, $\alpha_{i1}(x, \bar{s}_i) = \kappa[P(x, \mathbf{s})]^{-\gamma}$ and $\eta_{i1}(x, \bar{s}_i) = 0$ are used, where $P(x, \mathbf{s})$ in (2.37) indicates the extent to which point $x \in \Omega$ is covered by all the agents in $\mathcal{V}$. However, when evaluating the boosted gradient (2.45), $x \in V_i \subseteq \Omega$. Therefore, this approach assigns higher weights to points $x \in V_i$ that are less covered by the closed neighborhood $\bar{B}_i$.

**Neighbor-Boosting:** This boosting function family uses $\alpha_{i1}(x, \bar{s}_i) = 1$ and $\eta_{i1}(x, \bar{s}_i) = \sum_{j \in B_i} 1_{\{x = s_j\}} \cdot \frac{\kappa \cdot \mathbf{1}\{s_j \in V_i\}}{\|s_i - x\|^\gamma}$. As a result, agent $i$ gets repelled from the neighbors who are in its visibility region $V_i$.

## B.3 In Chapter 3

### B.3.1 Partial Curvature: A Brief Summary of (Liu et al., 2019) (used in Section 3.2.4)

The work in (Liu et al., 2018), proves that the partial curvature $\alpha_p$ in (3.8) always leads to a better performance bound compared to the total curvature $\alpha_t$ in (3.9), if the problem (3.1) satisfies a few additional conditions. Here, to understand these additional conditions, a brief summary of the findings of (Liu et al., 2019) is provided.

First, in addition to the Assumption 3.1, regarding the set-objective function $f$ in the problem (3.1), it is assumed that $f : \mathcal{I}_N \to \mathbb{R}$ (in contrast to taking $f : 2^X \to \mathbb{R}$).

**Domain Extension:** Consider the following definitions.

**Definition B.1.** *The **minor** of a set $A \subseteq X$ is a set denoted by $A^-$, where*

$$A^- \triangleq \underset{\substack{B:B\subset A,\\ |B|=|A|-1}}{\operatorname{argmax}} f(B). \tag{B.1}$$

**Definition B.2.** *A set function $f : \mathcal{I}_k \to \mathbb{R}$ is **extendable** to the domain $\mathcal{I}_{k+1}$ if there exist a function $g : \mathcal{I}_{k+1} \to \mathbb{R}$ (called the "extended" version of $f$) where,*

$$g(A) = \begin{cases} f(A), & A \in \mathcal{I}_k, \\ f(A^-) + d_{A,k}, & A \notin \mathcal{I}_k, \end{cases} \tag{B.2}$$

*for all $A \in \mathcal{I}_{k+1}$ with $d_{A,k} \in \mathbb{R}_{\geq 0}$ and $A^-$ is the minor of the set $A$.*

Due to the flexibility of selecting $d_{A,k}$, there exists an infinite number of extended versions (i.e., $g$) for a given set function $f$ between any two domains $\mathcal{I}_k$ to $\mathcal{I}_{k+1}$. Moreover, if the set-function $f$ is normalized or monotone, a corresponding extended set function $g$ will also inherit such properties. However, this is not generally true for the submodularity property unless $d_{A,k}$ values are chosen in the manner given below.

**Preserving the Submodularity Property:** If the set-objective $f$ is submodular, its extended version $g$ in (B.2) is also submodular is $d_{A,k}$ values are chosen such that

$$d_{A,k} \leq U_{A,k} \triangleq \min_{\substack{(B,a):|B|=k \\ a\in B\subset A}} \left[f(B) - f(B\backslash\{a\}) + f(A\backslash\{a\}) - f(A^{-})\right], \tag{B.3}$$

for all $A \subseteq X$ such that $|A| = k+1$. This condition is a result of applying the second submodularity condition in Def. 3.2 for the set function $g$ in (B.2).

Another interested property that needs to be preserved during an extension is: the existence of an extension $g$ such that $\alpha_p(f, \mathcal{I}_k) = \alpha_t(g, \mathcal{I}_{k+1})$. For convenience, henceforth, it is called as the *binding* property of $f$ between two domains $\mathcal{I}_k$ and $\mathcal{I}_{k+1}$.

**Preserving the Binding Property:** The extended version $g$ given in (B.2) preserves the aforementioned binding property if $d_{A,k}$ values are chosen such that

$$d_{A,k} \geq L_{A,k} \triangleq \max_{a:a\in A} \left[(1-\alpha_p(f,\mathcal{I}_k))f(\{a\}) + f(A\backslash\{a\}) - f(A^{-})\right], \tag{B.4}$$

for all $A \subseteq X$ such that $|A| = k+1$.

**Extending the Concept of Extension:** Note that (B.2), (B.3) and (B.4) are focused on extending the domain of a set-function $f$ from $\mathcal{I}_k$ to $\mathcal{I}_{k+1}$. Therefore, starting with $k = N$, one can repeatedly apply these steps to extend the domain of the set-objective function $f$ in (3.1) from $\mathcal{I}_N$ to $\mathcal{I}_n = 2^X$ (recall that $|X| = n$).

**Application of the Partial Curvature:** As stated in Theorem 3.2, to apply the partial curvature concept in an application, first, the existence of an extended version $g : \mathcal{I}_n \to \mathbb{R}$ of the set-objective function $f : \mathcal{I}_N \to \mathbb{R}$ (with preserved submodularity and binding properties), should be proven. For this purpose, the conditions given in (B.3) and (B.4) can be exploited. In particular, proving that $U_{A,k} - L_{A,k} \geq 0, \forall A \subseteq X$ such that $|A| = k+1$ for some generic $k \in \mathbb{Z}$ where $N \leq k \leq n$ is sufficient.

# Appendix C

# Appendices for Chapter 3

## C.1 Properties of the "Minor" of an Agent Set in Coverage

According to Def. B.1, the **minor** of a set $A \subseteq X$ with respect to the set-coverage function $H$ is a set denoted by $A^-$, where

$$A^- \triangleq \underset{\substack{B:B\subset A,\\ |B|=|A|-1}}{\operatorname{argmax}} H(B). \tag{C.1}$$

**Lemma C.1.** *The minor of a set of agents $A$ (i.e., $A^-$) can be obtained by removing the agent in $A$ with the lowest local objective function value, i.e., $A^- = A\backslash\{a^*\}$ with*

$$a^* = \underset{a\in A}{\operatorname{argmin}}\ \Delta H(a|A\backslash\{a\}).$$

*Proof.* See Appendix A.3.8. □

Thus, $A^-$ can be thought of as the remaining set of agents when the least contributing agent to set-coverage $H(A)$ is removed from the agent set $A$. The following three lemmas can now be established using basic properties of the coverage problem.

**Lemma C.2.** *For all $B, A$ such that $B \subset A \subseteq X$ and $|B| = |A| - 1$, for any $s_j \in B$,*

$$H(A^-) - H(A\backslash\{s_j\}) \leq H(B) - H(B\backslash\{s_j\}).$$

*Proof.* See Appendix A.3.9. □

This result is used to prove the extendability (introduced in Section B.3.1) of the set-coverage function $H(S)$, which enables the application of Theorem 3.2.

**Lemma C.3.** *For all $B, A$ such that $B \subset A \subseteq X$, $|B| = |A| - 1$ and $A^- \neq B$,*

$$H(A) - H(A^-) \leq H(B) - H(B^-).$$

*Proof.* See Appendix A.3.10. □

This lemma implies that the set-coverage function loss incurred when removing the worst contributing agent of a set $A \subseteq X$ is always smaller than that of a set $B \subset A$ such that $|B| = |A| - 1$, whenever $A^- \neq B$.

**Lemma C.4.** *If three sets $A_{k+1}, A_k, A_{k-1}$ are such that $A_{k+1}^- = A_k$, $A_k^- = A_{k-1}$, then*

$$H(A_{k+1}) - H(A_k) \leq H(A_k) - H(A_{k-1})$$

*Proof.* See Appendix A.3.11. □

This result indicates that if started with some set $A_{k+1}$ and iteratively removed the worst contributing agent, the loss in the set-coverage function would increase over such iterations. Next, lemmas C.3, C.4 are used to establish the following theorem.

**Theorem C.1.** *For all $B, A$ such that $B \subset A \subseteq X$,*

$$H(A) - H(A^-) \leq H(B) - H(B^-).$$

*Proof.* See Appendix A.3.12. □

This theorem generalizes the Lemma C.3 and shows that the coverage loss due to the removal of the worst contributing agent of any subset will be larger than that of any super-set. This result is also used to prove the applicability of Theorem 3.2 for the class of multi-agent coverage problems.

# Appendix D

# Appendices for Chapter 5

## D.1 Constrained Bivariate Rational Function Optimization

**Convexity of Rational Functions:** Consider a *rational function* $h : \mathbb{R} \to \mathbb{R}$ of the form $h(r) = \frac{f(r)}{g(r)}$ and assume $g(r) > 0 \ \forall r \in \mathcal{U} \subseteq \mathbb{R}$ where $\mathcal{U}$ is a closed interval. In the following, the argument of $f(r), g(r)$ or $h(r)$ is omitted for notational convenience. Also, the notation "$'$" is used to denote the derivative (with respect to $r$).

**Lemma D.1.** *Whenever polynomials $g(r)$ and $f(r)$ satisfy*

$$g[gf''' - fg'''] - 3g''[gf' - fg'] = 0, \ \forall r \in \mathcal{U}, \tag{D.1}$$

*$h(r)$ is convex (or concave) on $\mathcal{U}$ if $\Delta_h(r_0) > 0$ (or $\Delta_h(r_0) < 0$) where $r_0 \in \mathcal{U}$ and*

$$\Delta_h(r) \triangleq g[gf'' - fg''] - 2g'[gf' - fg']. \tag{D.2}$$

*Proof.* See Appendix A.5.4. □

**Remark D.1.** *According to Lemma D.1, the condition in* (D.1) *along with $\Delta_h(r_0) > 0$ (or $\Delta_h(r_0) < 0$) for some $r_0 \in \mathcal{U}$ is sufficient to determine the convexity (or concavity) of $h(r)$ on $\mathcal{U}$. As an example,* (D.1) *is satisfied whenever the rational function $h(r)$ has a denominator polynomial $g(r)$ of first degree and a numerator polynomial $f(r)$ of second degree. In such a case, the convexity/concavity of $h(r)$ over $\mathcal{U}$ can be identified by simply evaluating the sign of $\Delta_h(r)$ at a convenient $r = r_0 \in \mathcal{U}$ point.*

**Constrained Minimization of $h(r)$:** Assume $h(r) = \frac{f(r)}{g(r)}$ to be a rational function which satisfies the conditions discussed above: $g(r) > 0, \ \Delta_h'(r) = 0 \ \forall r \in \mathcal{U} \subseteq \mathbb{R}$. Further, assume the signs of $\Delta_h(r_0)$ and $h'(r_0)$ are known at some point of interest

$r = r_0 \in \mathcal{U}$ (recall that the sign of $\Delta_h(r_0)$ mimics the sign of $h''(r), r \in \mathcal{U}$). According to Lemma D.1, the latter assumption fully determines the convexity (or concavity) of $h(r)$ on $\mathcal{U}$ and its gradient direction at $r = r_0$, respectively. Now, consider the following optimization problem:

$$r^* = \underset{r_0 \leq r \leq r_1}{\operatorname{argmin}}\ h(r), \tag{D.3}$$

where $[r_0, r_1] \subseteq \mathcal{U}$. A critical $r$ value $r = r^{\#}$ (important to the analysis) is defined as

$$r^{\#} \triangleq \begin{cases} \{r : h'(r) = 0, r > r_0\} & \text{if } \Delta_h(r_0) > 0 \ \ \& \ \ h'(r_0) < 0 \\ \{r : h(r) = h(r_0), r > r_0\} & \text{if } \Delta_h(r_0) < 0 \ \ \& \ \ h'(r_0) > 0. \end{cases}$$

Note that the two cases considered above are the only ones where a stationary point of $h(r)$ could occur for some $r > r_0,\ r \in \mathcal{U}$ (see also Fig. D·1).

**Lemma D.2.** *The optimal solution to* (D.3) *is as follows:*

$$\textit{If } \Delta_h(r_0) < 0,\ h'(r_0) > 0, \qquad r^* = \begin{cases} r_1 & \textit{if } r_1 > r^{\#} \\ r_0 & \textit{otherwise,} \end{cases}$$

$$\textit{If } \Delta_h(r_0) > 0,\ h'(r_0) < 0, \qquad r^* = \begin{cases} r^{\#} & \textit{if } r_1 > r^{\#} \\ r_1 & \textit{otherwise,} \end{cases}$$

$$\textit{Otherwise,} \qquad r^* = \begin{cases} r_0 & \textit{if } \Delta_h(r_0) \geq 0 \\ & \textit{and } h'(r_0) \geq 0 \\ r_1 & \textit{otherwise.} \end{cases}$$

*Proof.* The proof easily follows by inspection of all cases shown in Fig. D·1. □

In essence, an optimization problem of the form (D.3) can be solved based exclusively on the values of $h'(r_0),\ \Delta_h(r_0)$ and $r^{\#}$. Note that $r^{\#}$ is only required in two special cases and for the application example mentioned in Remark D.1, it can be obtained simply by solving for the roots of a quadratic expression (single variable).

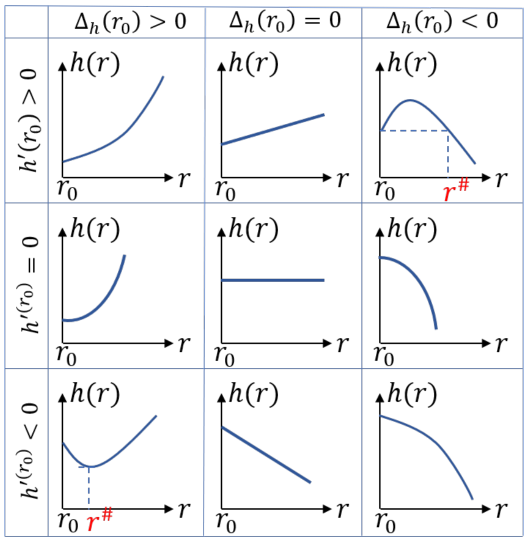


**Figure D·1:** Graphs of possible $\{h(r) : r \geq r_0,\ r \in \mathcal{U}\}$ profiles for different cases of $h'(r_0)$ and $\Delta_h(r_0)$ (recall $sgn(\Delta_h(r_0)) = sgn(h''(r))$ determines the convexity or concavity).

**Bivariate Rational Functions:** Next, consider the class of *bivariate rational functions* that can be represented by a function $H : \mathbb{R}^2_+ \to \mathbb{R}$ of the form

$$H(x,y) = \frac{F(x,y)}{G(x,y)} = \frac{C_1x^2 + C_2y^2 + C_3xy + C_4x + C_5y + C_6}{C_7x + C_8y + C_9}, \tag{D.4}$$

where the coefficients $C_1, \ldots, C_9$ are known scalar constants with $C_7 \geq 0,\ C_8 \geq 0$ and $C_9 > 0$. Also $\mathbb{R}^2_+$ denotes the non-negative orthant of $\mathbb{R}^2$.

Developing conditions for the convexity of $H(x,y)$ is a complicated task. Even if such conditions were derived, interpreting them and exploiting them to solve a two-dimensional constrained optimization problem that involves minimizing $H(x,y)$ (analogous to (D.3)) is challenging. To address this, the behavior of $H(x,y)$ is next studied along a generic line segment of the form $y = mx + b$ starting at some point $(x_0, y_0) \in \mathbb{R}^2_+$ as shown in Fig. D·2a. A parameter $r$ is used to represent a generic location $(x_r, y_r)$ on this line as $(x_r, y_r) = (x_0 + r,\ y_0 + mr)$ where $r$ is introduced exploiting the gradient $m$ of the line segment:

$$\frac{y_r - y_0}{x_r - x_0} = m \implies \frac{y_r - y_0}{m} = \frac{x_r - x_0}{1} = r. \tag{D.5}$$

A rational function $h(r)$ can now be defined as

$$h(r) \triangleq H(x_0 + r,\ y_0 + mr) = \frac{F(x_0 + r,\ y_0 + mr)}{G(x_0 + r,\ y_0 + mr)} = \frac{f(r)}{g(r)}, \tag{D.6}$$

to represent $H(x, y)$ along the line segment of interest.

The parameter $r$ is constrained such that $r \in \mathcal{U} \triangleq [-x_0, \frac{-y_0}{m}]$ to limit the line segment to $\mathbb{R}^2_+$. This allows $h(r)$ to fall directly into the category of rational functions discussed in Lemma D.1 and in Remark D.1.

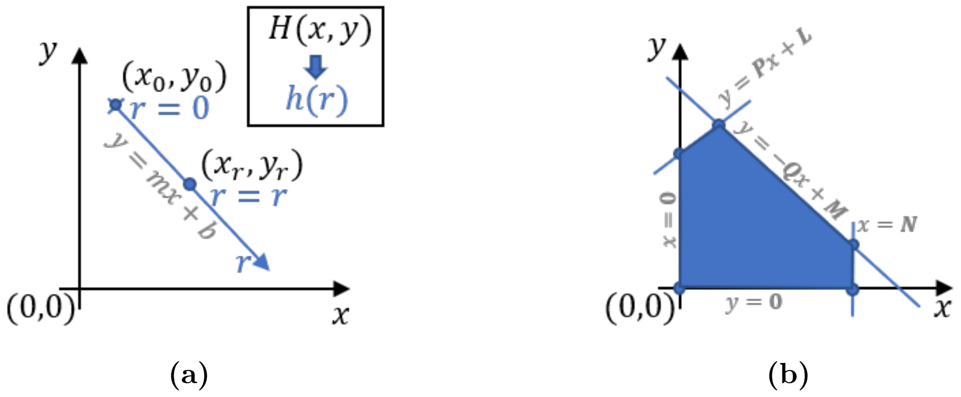


**Figure D·2:** (a) $H(x, y)$ along the line $y = mx + b$, (b) Feasible space for $H(x, y)$ in (D.8).

**Theorem D.1.** *The rational function $h(r)$, $r \in \mathcal{U}$ defined in* (D.6) *is convex (or concave) if $\Delta_h(r_0) > 0$ (or $\Delta_h(r_0) < 0$), where $r_0 \in \mathcal{U}$ and $\Delta_h(r)$ is defined in* (D.2).

*Proof.* According to (D.6) and $\mathcal{U}$ defined above, the denominator polynomial $g(r) = G(x_0 + r, y_0 + mr) > 0$ for all $r \in \mathcal{U}$ as $C_7 \geq 0$, $C_8 \geq 0$ and $C_9 > 0$ in (D.4).

Since $g(r)$ and $f(r)$ are polynomials of degree 1 and 2 respectively, they satisfy condition (D.1). Thus, Lemma D.1 is applicable for $h(r)$ in (D.6) and its convexity will depend on the condition $\Delta_h(r_0) > 0$. □

It is worth pointing out that $\Delta_h(r)$ is in fact independent of $r$ as $\Delta_h'(r) = 0, \forall r \in \mathcal{U}$ (see the last step of the proof of Lemma D.1 and (D.1)). However, it will depend on other parameters contained in (D.4) including $x_0, y_0$ and $m$. For example, when the line segment defined by $x_0 = 0, y_0 = 0, m = 0$ (i.e., the $x$-axis) is used, $\Delta_h(r) = 2C_6C_7^2 - 2C_4C_7C_9 + 2C_1C_9^2, \forall r \in \mathbb{R}_{\geq 0}$.

In the introduced parameterization scheme, the parameter $r$ represents the distance along the $x$ axis from $x_0$ (projected from the line segment $y = mx+b$). However, if $H(x, y)$ needs to be studied along the $y$ axis (from $y_0$ projected from a line segment $x = ny + c$), then using

$$\frac{y_r - y_0}{x_r - x_0} = \frac{1}{n} \implies \frac{y_r - y_0}{1} = \frac{x_r - x_0}{n} = r, \tag{D.7}$$

is more appropriate as it gives $(x_r, y_r) = (x_0 + nr, y_0 + r)$.

Theorem D.1 enables determining the optimal $H(x, y)$ value along a known line segment (on $\mathbb{R}_+^2$) using Lemma D.2 for a problem of the form (D.3). This capability is exploited next.

**Constrained Minimization of $H(x, y)$:** The main objective of this discussion is to obtain a closed form solution to a constrained optimization problem of the form

$$\begin{aligned} (x^*, y^*) = \underset{(x,y)}{\text{argmin}}\ & H(x, y) \\ & 0 \leq x \leq \mathbf{N}, \\ & 0 \leq y \leq \min\{\mathbf{P}x + \mathbf{L},\ -\mathbf{Q}x + \mathbf{M}\}, \end{aligned} \tag{D.8}$$

where $H(x, y)$ is a known bivariate rational function of the form (D.4) and $\mathbf{P}, \mathbf{Q}, \mathbf{L}, \mathbf{M}$ are known positive (scalar) constants. These constraints define a convex 2-Polytope as shown in Fig. D·2b. The steps to solve the above problem are discussed next.

**- Step 1:** The unconstrained version of (D.8) is considered first. This is solved using the KKT necessary conditions (Bertsekas, 2016), which reveal two equations of generic conics (Rosenberg, 2010). Therefore, the stationary points of $H(x, y)$ lie at the (four) intersection points of those two conics. The problem of determining the intersection of two conics boils down to solving a quartic equation, which has a well-known closed-form solution (Auckly, 2007). These (four) solutions are computed and stored in a *solution pool* if they satisfy the problem constraints.

**- Step 2:** Next, the constrained version of (D.8) is considered. In such a case, it is possible for $(x^*, y^*)$ to lie on a constraint boundary. To capture such situations, $H(x, y)$ is optimized along each of the boundary line segments of the feasible space (there are five of them as shown in Fig. D·2b).

On a selected boundary line segment, the first step is to parameterize $H(x, y)$ to obtain a single variable rational function $h(r)$ (following either (D.5) or (D.7)). Then, the next step is to solve the resulting convex (or concave) optimization problem (of the form (D.3)) using Lemma D.2. Note that this is enabled by Theorem D.1. Finally, the obtained optimal solution is added to the solution pool from **Step 1**.

**- Step 3:** The final step is to pick the best solution out of the solution pool (which only contains at most nine candidates solutions). Therefore, this is achieved by directly evaluating $H(x, y)$ and comparing all candidate solutions to each other.

This approach is computationally cheap, accurate and provides the global optimal solution compared to gradient-based methods which are susceptible to local optima. This concludes the discussion on how to solve a generic problem of the form (D.8).

# References

Addis, B., Locatelli, M., and Schoen, F. (2005). Local Optima Smoothing for Global Optimization. *Optimization Methods and Software*, 20(4-5):417–437.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows : Theory, Algorithms, and Applications*. Prentice Hall.

Aksaray, D., Leahy, K., and Belta, C. (2015). Distributed Multi-Agent Persistent Surveillance Under Temporal Logic Constraints. *IFAC-PapersOnLine*, 48(22):174–179.

Anagnostopoulos, T., Zaslavsky, A., Sosunova, I., Fedchenkov, P., Medvedev, A., Ntalianis, K., Skourlas, C., Rybin, A., and Khoruznikov, S. (2018). A Stochastic Multi-Agent System for Internet of Things-Enabled Waste Management in Smart Cities. *Waste Management and Research*, 36(11):1113–1121.

Arora, J. S., Elwakeil, O. A., Chahande, A. I., and Hsieh, C. C. (1995). Global Optimization Methods for Engineering Applications: A Review. *Structural Optimization*, 9(3-4):137–159.

Auckly, D. (2007). Solving the Quartic with a Pencil. *American Mathematical Monthly*, 114(1):29–39.

Bastianello, N., Carli, R., Schenato, L., and Todescato, M. (2018). A Partition-Based Implementation of the Relaxed ADMM for Distributed Convex Optimization over Lossy Networks. In *Proceedings of 57th IEEE Conference on Decision and Control*, pages 3379–3384.

Bektas, T. (2006). The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures. *Omega*, 34(3):209–219.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy Layer-wise Training of Deep Networks. In *Proceedings of Advances in neural information processing systems*, pages 153–160.

Bentz, W. and Panagou, D. (2018). Energy-aware Persistent Coverage and Intruder Interception in 3D Dynamic Environments. In *Proceedings of American Control Conference*, volume 2018-June, pages 4426–4433.

Bertsekas, D. P. (2016). *Nonlinear Programming*. Athena Scientific.

Blazinskas, A. and Misevicius, A. (2011). Combining 2-Opt, 3-Opt and 4-Opt with K-Swap-Kick Perturbations for the Travelling Salesman Problem. Technical report. Available at https://api.semanticscholar.org/CorpusID:15324387 [Verified 18 March 2021].

Blum, C. (2005). Ant Colony Optimization: Introduction and Recent Trends. *Physics of Life Reviews*, 2(4):353–373.

Bof, N., Carli, R., and Schenato, L. (2018). Lyapunov Theory for Discrete Time Systems. *arXiv e-prints*, page 1809.05289.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Caprari, G., Breitenmoser, A., Fischer, W., Hürzeler, C., Tâche, F., Siegwart, R., Schoeneich, P., Rochat, F., Mondada, F., and Moser, R. (2010). Highly Compact Robots for Inspection of Power Plants. In *International Conference on Applied Robotics for the Power Industry.*

Cassandras, C. G. and Lafortune, S. (2010). *Introduction to Discrete Event Systems.* Springer Publishing Company, Inc., 2nd edition.

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.

Chen, R. and Cassandras, C. G. (2020a). Optimal Assignments in Mobility-on-Demand Systems Using Event-Driven Receding Horizon Control. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15.

Chen, R. and Cassandras, C. G. (2020b). Optimization of Ride Sharing Systems Using Event-driven Receding Horizon Control. In *Proceedings of 2020 International Workshop on Discrete Event Systems (to appear).*

Chiu, P. and Lin, F. (2004). A Simulated Annealing Algorithm to Support the Sensor Placement for Target Location. In *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, pages 867–870.

Conforti, M. and Cornuéjols, G. (1984). Submodular Set Functions, Matroids and The Greedy Algorithm: Tight Worst-Case Bounds and Some Generalizations of The Rado-Edmonds Theorem. *Discrete Applied Mathematics*, 7(3):251–274.

Dai, L., Cao, Q., Xia, Y., and Gao, Y. (2017). Distributed MPC for Formation of Multi-Agent Systems with Collision Avoidance and Obstacle Avoidance. *Journal of the Franklin Institute*, 354(4):2068–2085.

Davis, L. (1996). *Handbook of Genetic Algorithms.* London International Thomson Computer Press, Boston.

Dotoli, M., Hammadi, S., Jeribi, K., Russo, C., and Zgaya, H. (2013). A Multi-Agent Decision Support System for Optimization of Co-Modal Transportation Route Planning Services. In *Proceedings of 52nd IEEE Conference on Decision and Control*, pages 911–916.

Fan, F., Wu, G., Wang, M., Cao, Q., and Yang, S. (2018). Multi-Robot Cyber Physical System for Sensing Environmental Variables of Transmission Line. *Sensors (Switzerland)*, 18(9).

Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. (1978). An Analysis of Approximations for Maximizing Submodular Set Functions—II. *Polyhedral Combinatorics: Dedicated to the memory of D.R. Fulkerson*, pages 73–87.

Flanders, H. (1973). Differentiation Under the Integral Sign. *The American Mathematical Monthly*, 80(6):615–627.

Floudas, C. A. and Gounaris, C. E. (2008). A Review of Recent Advances in Global Optimization. *Journal of Global Optimization*, 45(1):3–38.

Floudas, C. A., Pardalos, P. M., Adjiman, C. S., Esposito, W. R., Gümüş, Z. H., Harding, S. T., Klepeis, J. L., Meyer, C. A., and Schweiger, C. A. (1999). Dynamic Optimization Problems BT - Handbook of Test Problems in Local and Global Optimization. pages 351–412. Springer US.

Ghapani, S., Mei, J., Ren, W., and Song, Y. (2016). Fully Distributed Flocking with A Moving Leader for Lagrange Networks with Parametric Uncertainties. *Automatica*, 67:67–76.

Hari, S. K., Rathinam, S., Darbha, S., Kalyanam, K., Manyam, S. G., and Casbeer, D. (2019). The Generalized Persistent Monitoring Problem. In *Proceedings of American Control Conference*, volume 2019-July, pages 2783–2788.

Hari, S. K. K., Rathinam, S., Darbha, S., Kalyanam, K., Manyam, S. G., and Casbeer, D. (2018). Persistent Monitoring of Dynamically Changing Environments Using an Unmanned Vehicle. *arXiv e-prints*, page 1808.02545.

Holland, J. H. (1984). Genetic Algorithms and Adaptation. In *Adaptive Control of Ill-Defined Systems*, pages 317–333. Springer US.

Hoos, H. H. and Stutzle, T. (2005). Praise for Stochastic Local Search: Foundations and Applications. In *Stochastic Local Search*, pages i–ii. Morgan Kaufmann Publishers.

Huynh, V. A., Enright, J. J., and Frazzoli, E. (2010). Persistent Patrol with Limited-Range On-Board Sensors. In *Proceedings of 49th IEEE Conference on Decision and Control*, pages 7661–7668.

Ilie, S. and Bădică, C. (2013). Multi-Agent Approach to Distributed Ant Colony Optimization. In *Proceedings of Science of Computer Programming*, volume 78, pages 762–774.

Jianbo Shi and Malik, J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948.

Khazaeni, Y. and Cassandras, C. G. (2018a). Event-Driven Cooperative Receding Horizon Control for Multi-Agent Systems in Uncertain Environments. *IEEE Transactions on Control of Network Systems*, 5(1):409–422.

Khazaeni, Y. and Cassandras, C. G. (2018b). Event-Driven Trajectory Optimization for Data Harvesting in Multi-Agent Systems. *IEEE Transactions on Control of Network Systems*, 5(3):1335–1348.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Lan, X. and Schwager, M. (2013). Planning Periodic Persistent Monitoring Trajectories for Sensing Robots in Gaussian Random Fields. In *In Proceedings of IEEE International Conference on Robotics and Automation*, pages 2415–2420.

Lasdon, L. and Plummer, J. C. (2008). Multistart algorithms for seeking feasibility. *Computers and Operations Research*, 35(5):1379–1393.

Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. (2017). First-Order Methods Almost Always Avoid Saddle Points. *Mathematical Programming*, 176(1-2):311–337.

Li, W. and Cassandras, C. G. (2006). A Cooperative Receding Horizon Controller for Multi-Vehicle Uncertain Environments. *IEEE Transactions on Automatic Control*, 51(2):242–257.

Li, X. and Liu, H. (2018). Greedy Optimization for K-Means-Based Consensus Clustering. *Tsinghua Science and Technology*, 23(2):184–194.

Liaqat, A., Hutabarat, W., Tiwari, D., Tinkler, L., Harra, D., Morgan, B., Taylor, A., Lu, T., and Tiwari, A. (2019). Autonomous Mobile Robots in Manufacturing: Highway Code Development, Simulation and Testing. *International Journal of Advanced Manufacturing Technology*, 104(9-12):4617–4628.

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.

Lin, Z., Wang, L., Han, Z., and Fu, M. (2014). Distributed Formation Control of Multi-Agent Systems Using Complex Laplacian. *IEEE Transactions on Automatic Control*, 59(7):1765–1777.

Lindemann, L. and Dimarogonas, D. V. (2019). Control Barrier Functions for Multi-Agent Systems Under Conflicting Local Signal Temporal Logic Tasks. *IEEE Control Systems Letters*, 3(3):757–762.

Liu, Y., Chong, E. K. P., and Pezeshki, A. (2018). Improved Bounds for The Greedy Strategy in Optimization Problems with Curvature. *Journal of Combinatorial Optimization*, 37(4):1126–1149.

Liu, Y., Chong, E. K. P., Pezeshki, A., and Zhang, Z. (2019). Submodular Optimization Problems and Greedy Strategies: A Survey. *arXiv e-prints*, page 1905.03308.

Maini, P., Yu, K., Sujit, P. B., and Tokekar, P. (2018). Persistent Monitoring with Refueling on a Terrain Using a Team of Aerial and Ground Robots. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pages 8493–8498.

Marden, J. R. and Roughgarden, T. (2014). Generalized Efficiency Bounds in Distributed Resource Allocation. *IEEE Transactions on Automatic Control*, 59(3):571–584.

Maza, I., Caballero, F., Capitán, J., Martínez-De-Dios, J. R., and Ollero, A. (2011). Experimental Results in Multi-UAV Coordination for Disaster Management and Civil Security Applications. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 61(1-4):563–585.

Menendez, O., Auat Cheein, F. A., Perez, M., and Kouro, S. (2017). Robotics in Power Systems: Enabling a More Reliable and Safe Grid. *IEEE Industrial Electronics Magazine*, 11(2):22–34.

Miller, K. S. (1981). On the Inverse of the Sum of Matrices. *Mathematics Magazine*, 54(2):67.

Molzahn, D. K., Dorfler, F., Sandberg, H., Low, S. H., Chakrabarti, S., Baldick, R., and Lavaei, J. (2017). A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems. *IEEE Transactions on Smart Grid*, 8(6):2941–2962.

Nedić, A. and Liu, J. (2018). Distributed Optimization for Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):77–103.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. (1978). An Analysis of Approximations for Maximizing Submodular Set Functions—I. *Mathematical Programming*, 14(1):265–294.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On Spectral Clustering: Analysis and an Algorithm. In *Proceedings of 14th International Conference on Neural Information Processing Systems*, pages 849–856.

Nilsson, C. (2003). Heuristics for the Traveling Salesman Problem. Technical report, Linkoping University. Available at https://www.researchgate.net/publication/228906083_Heuristics_for_the_Traveling_Salesman_Problem [Verified 16 April 2021].

Panageas, I., Piliouras, G., and Wang, X. (2019). First-Order Methods Almost Always Avoid Saddle Points: The Case of Vanishing Step-Sizes. *arXiv e-prints*, page 1906.07772.

Pinto, S. C., Andersson, S. B., Hendrickx, J. M., and Cassandras, C. G. (2019). Optimal Multi-Agent Persistent Monitoring of the Uncertain State of a Finite Set of Targets. In *Proceedings of 58th IEEE Conference on Decision and Control*, volume 2019-Decem, pages 4280–4285.

Pinto, S. C., Andersson, S. B., Hendrickx, J. M., and Cassandras, C. G. (2020a). Multi-Agent Infinite Horizon Persistent Monitoring of Targets with Uncertain States in Multi-Dimensional Environments. In *Proceedings of 21st IFAC World Congress*.

Pinto, S. C., Andersson, S. B., Hendrickx, J. M., and Cassandras, C. G. (2020b). Optimal Minimax Mobile Sensor Scheduling Over a Network. *arXiv e-prints*, page 2009.11386.

Rezazadeh, N. and Kia, S. S. (2019). A Sub-Modular Receding Horizon Approach to Persistent Monitoring for A Group of Mobile Agents Over an Urban Area. In *IFAC-PapersOnLine*, volume 52, pages 217–222.

Rosenberg, S. (2010). Conics. Technical report. Available at http://math.bu.edu/people/sr/GandS/handouts/Chapter6.pdf [Verified 18 March 2021].

Schwager, M., Bullo, F., Skelly, D., and Rus, D. (2008). A Ladybug Exploration Strategy for Distributed Adaptive Coverage Control. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2346–2353.

Shen, Z. and Andersson, S. B. (2011). Tracking Nanometer-Scale Fluorescent Particles in Two Dimensions with a Confocal Microscope. *IEEE Transactions on Control Systems Technology*, 19(5):1269–1278.

Smith, S. L., Schwager, M., and Rus, D. (2011). Persistent Monitoring of Changing Environments Using a Robot with Limited Range Sensing. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 5448–5455.

Song, C., Liu, L., Feng, G., and Xu, S. (2014). Optimal Control for Multi-Agent Persistent Monitoring. *Automatica*, 50(6):1663–1668.

Sun, C., Welikala, S., and Cassandras, C. G. (2020). Optimal Composition of Heterogeneous Multi-Agent Teams for Coverage Problems with Performance Bound Guarantees. *Automatica*, 117:108961.

Sun, H., Liu, Y., Li, F., and Niu, X. (2017a). A Survey On Optimal Consensus of Multi-Agent Systems. In *Proceedings of IEEE Chinese Automation Congress*, pages 4978–4983.

Sun, X. and Cassandras, C. G. (2016). Optimal Dynamic Formation Control of Multi-Agent Systems in Constrained Environments. *Automatica*, 73:169–179.

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 *Proceedings of 53rd IEEE Conference on Decision and Control*, pages 3701–3706.

Sun, X., Cassandras, C. G., and Meng, X. (2017b). A Submodularity-Based Approach for Multi-Agent Optimal Coverage Problems. In *Proceedings of 56th IEEE Conference on Decision and Control*, pages 4082–4087.

Sun, X., Cassandras, C. G., and Meng, X. (2019). Exploiting Submodularity to Quantify Near-Optimality in Multi-Agent Coverage Problems. *Automatica*, 100:349–359.

Tfaili, W. and Siarry, P. (2008). A New Charged Ant Colony Algorithm for Continuous Dynamic Optimization. *Applied Mathematics and Computation*, 197(2):604–613.

Trevathan, J. and Johnstone, R. (2018). Smart Environmental Monitoring and Assessment Technologies (SEMAT)—A New Paradigm for Low-Cost, Remote Aquatic Environmental Monitoring. *Sensors (Switzerland)*, 18(7).

Valenti, M. J. (2007). *Approximate Dynamic Programming with Applications in Multi-agent Systems*. PhD thesis, Massachusetts Institute of Technology.

von Luxburg, U. (2007). A Tutorial on Spectral Clustering. *arXiv e-prints*, page 0711.0189.

Wang, L., Liu, J., and Morse, A. S. (2019). A Distributed Observer for a Continuous-Time Linear System. In *Proceedings of American Control Conference*, pages 86–91.

Wang, X., Yi, P., and Hong, Y. (2014). Dynamic Optimization for Multi-Agent Systems with External Disturbances. *Control Theory and Technology*, 12(2):132–138.

Wang, Y.-W., Wei, Y.-W., Liu, X.-K., Zhou, N., and Cassandras, C. G. (2017). Optimal Persistent Monitoring Using Second-Order Agents with Physical Constraints. *IEEE Transactions on Automatic Control*, 64(8):3239–3252.

Wang, Z., Moran, B., Wang, X., and Pan, Q. (2016). Approximation for Maximizing Monotone Non-Decreasing Set Functions with A Greedy Method. *Journal of Combinatorial Optimization*, 31(1):29–43.

Welikala, S. and Cassandras, C. G. (2019a). Asymptotic Analysis for Greedy Initialization of Threshold-Based Distributed Optimization of Persistent Monitoring on Graphs. *arXiv e-prints*, page 1911.02658.

Welikala, S. and Cassandras, C. G. (2019b). Distributed Non-Convex Optimization of Multi-Agent Systems Using Boosting Functions to Escape Local Optima: Theory and Applications. *arXiv e-prints*, page 1903.04133.

Welikala, S. and Cassandras, C. G. (2020a). Distributed Non-Convex Optimization of Multi-Agent Systems Using Boosting Functions to Escape Local Optima. *IEEE Transactions on Automatic Control*.

Welikala, S. and Cassandras, C. G. (2020b). Event-Driven Receding Horizon Control for Distributed Estimation in Network Systems. *arXiv e-prints*, page 2009.11958.

Welikala, S. and Cassandras, C. G. (2020c). Event-Driven Receding Horizon Control For Distributed Persistent Monitoring in Network Systems. *arXiv e-prints*, page 2003.11713.

Welikala, S. and Cassandras, C. G. (2021a). Event-Driven Receding Horizon Control for Distributed Estimation in Network Systems. In *Proceedings of American Control Conference (to appear)*.

Welikala, S. and Cassandras, C. G. (2021b). Event-Driven Receding Horizon Control For Distributed Persistent Monitoring in Network Systems. *Automatica*, 127:109519.

Welikala, S. and Cassandras, C. G. (2021c). Event-Driven Receding Horizon Control of Energy-Aware Dynamic Agents for Distributed Persistent Monitoring. *arXiv e-prints*, page 2102.12963.

Welikala, S. and Cassandras, C. G. (2021d). Event-Driven Receding Horizon Control of Energy-Aware Dynamic Agents for Distributed Persistent Monitoring. In *Proceedings of 60th IEEE Conference on Decision and Control (submitted)*.

Xiao, W., Belta, C., and Cassandras, C. G. (2019). Decentralized Merging Control in Traffic Networks: A Control Barrier Function Approach. In *Proceedings of 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 270–279.

Xie, X. F. and Liu, J. (2009). Multi-Agent Optimization System for Solving the Traveling Salesman Problem (TSP). *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):489–502.

Xu, J., Tekin, C., Zhang, S., and van der Schaar, M. (2015). Distributed Multi-Agent Online Learning Based on Global Feedback. *IEEE Transactions on Signal Processing*, 63(9):2225–2238.

Yamashita, A., Arai, T., Ota, J., and Asama, H. (2003). Motion Planning of Multiple Mobile Robots for Cooperative Manipulation and Transportation. *IEEE Transactions on Robotics and Automation*, 19(2):223–237.

Yao, C., Ding, X. C., and Cassandras, C. G. (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.

Yazdani, D., Branke, J., Omidvar, M. N., Nguyen, T. T., and Yao, X. (2018). Changing or Keeping Solutions in Dynamic Optimization Problems with Switching Costs. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1095–1102.

Yu, J., Karaman, S., and Rus, D. (2015). Persistent Monitoring of Events With Stochastic Arrivals at Multiple Stations. *IEEE Transactions on Robotics*, 31(3):521–535.

Yu, J., Schwager, M., and Rus, D. (2016). Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks. *IEEE Transactions on Robotics*, 32(5):1106–1118.

Zhong, M. and Cassandras, C. G. (2011). Distributed Coverage Control and Data Collection with Mobile Sensor Networks. *IEEE Transactions on Automatic Control*, 56(10):2445–2455.

Zhou, N., Cassandras, C. G., Yu, X., and Andersson, S. B. (2019). Optimal Threshold-Based Distributed Control Policies for Persistent Monitoring on Graphs. In *Proceedings of American Control Conference*, pages 2030–2035.

Zhou, N., Yu, X., Andersson, S. B., and Cassandras, C. G. (2018). Optimal Event-Driven Multi-Agent Persistent Monitoring of a Finite Set of Data Sources. *IEEE Transactions on Automatic Control*, 63(12):4204–4217.

Zhu, M. and Martínez, S. (2013). An Approximate Dual Subgradient Algorithm for Multi-Agent Non-Convex Optimization. *IEEE Transactions on Automatic Control*, 58(6):1534–1539.

# CURRICULUM VITAE

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]