UNIVERSITY OF CALIFORNIA
RIVERSIDE

Distributed Optimization in Multi-Agent Systems: Continuous-Time Convex Optimization and Policy Optimization Based Packet Routing

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Shan Sun

June 2021

The Dissertation of Shan Sun is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to express my deepest gratitude to my adviser, Dr. Wei Ren, for his constant guidance, support, and encouragement through each stage of my Ph.D. career. It has been a great privilege to work with him and to learn from him over the past four years. He has made great achievements in academia, yet he is one of the kindest and most humble men I have ever known. Since met him, I even believe that the most successful people are usually the most humble ones. He has set a great example that will have a lifetime influence on my attitude and behavior towards work and life. I feel extremely lucky to be his student. I would also like to thank my mentors, Dr. Fei Chen, Dr. Jay A. Farrell, and Dr. Mariam Kiran for their valuable suggestions and advice for my research. The discussions with them help me a lot. Many thanks to other members in my oral and defense committee, Dr. Matt Barth, Dr. Jiasi Chen, Dr. Konstantinos Karydis, and Dr. Fabio Pasqualetti. I appreciate all their useful suggestions and insightful comments to enhance the quality of this work.

I also want to thank all my wonderful colleagues, Yong Ding, Ryan Decker, Qianjun Liu, Shaoshu Su, Dr. Peng Wang, Jie Xu, Pengxiang Zhu, Yifan Zhang, and all the visiting scholars in COVEN lab, for much joyful time together as well as many useful discussions, suggestions and encouragements. Meanwhile, I want to thank to my friends, Yue Chang, Mengfu Di, Xuan Gong, Shuai Huang, Zeyi Jiang, Xinyue Kan, Junying Liu, Runze Li, Shasha Li, Zhichao Liu, Zhilu Liu, Zhouyu Lu, Dr. Bashir Mohammed, Ashim Neupane, Lu Shi, Hanzhe Teng, An Xin, Fangfang Yang, Dr. Luting Yang, Bohan Zhao, for making my graduate life colorful, enriched, and memorable. I wish you all the best in your future endeavors.

I owe my deepest gratitude to my family. I am indebted to my parents, Jinggang Sun and Shengmei Sun, 's unreserved love and care. Thank my husband Dr. Hongsheng Yu, for not only accompanying me during my whole Ph.D. life but also providing technical suggestions whenever I encountered difficulties in research.

This work was supported by National Science Foundation under Grant ECCS-1920798 and Grant ECCS-1611423. Acknowledge of previously materials: the text of this dissertation, is partly rewritten based on the material that presented in seven previously published or submitted papers. The papers are as follows.

1, S. Sun, F. Chen, and W. Ren, Distributed average tracking over weight-unbalanced directed graphs, American Control Conference, 2019.

2, S. Sun, F. Chen, and W. Ren, Distributed average tracking in weight-unbalanced directed networks, IEEE Transactions on Automatic Control, 2020, DOI: 10.1109/TAC.2020.3046029.

3, S. Sun, Y. Zhang, P. Lin, W. Ren, and J. A. Farrell, Distributed time-varying optimization with state-dependent gains: algorithms and experiments, IEEE Transactions on Control Systems Technology, 2021, DOI: 10.1109/TCST.2021.3058845.

4, S. Sun, W. Ren, Distributed continuous-time optimization with time-varying objective functions and inequality constraints, IEEE Conference on Decision and Control, 2020.

5, S. Sun, J. Xu, and W. Ren, Distributed continuous-time algorithms for time-varying constrained convex optimization, IEEE Transactions on Automatic Control, under review.

6, S. Sun, M. Kiran, Multi-agent meta reinforcement learning for packet routing in dynamic network environments, ACM/IEEE Supercomputing Conference, 2020.

7, S. Sun, M. Kiran, and W. Ren, MAMRL: Exploiting multi-agent meta reinforcement

learning in WAN traffic engineering, ACM SIGCOMM Computer Communication Review, under review.

Dedicated to my parents,

Jinggang and Shengmei,

and to my husband, Hongsheng.

Anything good that has come to my life has been because of your love.

ABSTRACT OF THE DISSERTATION

Distributed Optimization in Multi-Agent Systems: Continuous-Time Convex Optimization and Policy Optimization Based Packet Routing

by

Shan Sun

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, June 2021
Dr. Wei Ren, Chairperson

A multi-agent system is defined as a collection of intelligent agents which are able to interact with each other or with their environments to solve problems that are beyond the individual capacities or knowledge of each problem solver. Distributed optimization in multi-agent system allows for acting the agents in a distributed manner (using only local information from their neighbors) to achieve global optimization objectives cooperatively so as to increase flexibility and robustness. Examples of cooperative tasks include optimization of network flows, big-data analysis, design of sensor networks, multi-robot teams, and resource allocation. In this dissertation, two distributed optimization problems are investigated, where two different methods of convex optimization and reinforcement learning are employed to solve these problems.

In our first problem, a distributed time-varying convex optimization problem is studied for continuous-time multi-agent systems. The objective is to minimize the sum of local time-varying objective functions, each of which is known to only an individual agent, through local interaction. Here, the optimal point is time varying and creates an optimal trajectory, which renders extra challenge to well-studied distributed time-invariant optimization problems. To begin with, we study distributed time-varying optimization problems with convex objective functions for undirected topologies. We propose multiple optimization algorithms for different application scenarios: 1) optimization problems that do not have explicit bounds on any information about the local objective functions, 2) optimization problems with linear equality constraints, 3) optimization problems with nonlinear inequality constraints, and 4) optimization problems with both linear equality and nonlinear inequality constraints. Furthermore, we aim to seek a design methodology for distributed time-varying optimization under possibly weight-unbalanced directed networks—the most general and thus most challenging case from the network topology perspective. Particularly, we study distributed time-varying optimization problems with quadratic objective functions, which are equivalent to distributed average tracking problems.

In the last part of this dissertation, the packet routing (path optimization) problem is addressed for distributed communication network systems that are consist of multiple routers and multiple links. Here, the goal is that all the routers work cooperatively to get all the packets delivered to their destinations through available paths with minimum time overall. Specifically, we aim to find optimal paths for packets in the presence of link failures, which is a crucial challenge faced by communication networks. We propose to leverage deep policy optimization (reinforcement learning) algorithms for enabling distributed model-free control in communication networks and present a novel meta-learning-based framework for enabling quick adaptation to topology changes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In this dissertation, we study optimization problems in multi-agent systems. A multi-agent system is defined as a collection of intelligent agents which are able to interact with each other or with their environments to solve desirable global objectives cooperatively. Applications for multi-agent systems include but are not limited to power systems, communication networks, smart buildings, and networked robotics. There are two commonly-used approaches in multi-agent technologies: the centralized approach and the distributed (decentralized) approach. In the centralized approach, there is at least one central agent that collects all of the information from other agents to make decisions for the whole group of agents. On the contrary, in the distributed approach, each agent only uses local information to achieve the desirable global objective. Compared to the centralized approach, the major motivations of using distributed approaches include, 1) many real world problems are considered distributed systems by nature and not by choice. Such systems cannot be dealt with by centralized strategies, 2) distributed systems are more tolerant to faults (more robust). Crashes in a distributed system may cause less degradation on the performance since the agents work on local information. Many problems in multi-agent systems can be posed in the framework of convex optimization [13]. In this dissertation, we consider two optimization problems in multi-agent systems: continuous-time convex optimization with time-varying objective functions and packet routing (path optimization) problems for communication networks. More specifically:

- First, we consider the distributed time-varying optimization with general convex objective functions for undirected topologies, where the goal is to minimize the sum of local time-varying objective functions.

- Next, we consider a special case of distributed time-varying optimization for possibly unbalanced graphs: distributed average tracking problems, where each objective function is quadratic.

- Finally, we consider packet routing problem in communication networks that are consist of multiple routers and multiple links. We aim to find optimal paths for the packets, through which the packets can be delivered to their destinations with minimum time.

1.2 Related Literature

In this section, we present literature review on three topics: distributed time-varying optimization with general convex objective functions, distributed time-varying optimization with quadratic objective functions (distributed average tracking) and packet routing.

1.2.1 Distributed Convex Optimization

There has been significant attention on distributed convex optimization problems, where the goal is to cooperatively seek the optimal solution that minimizes the sum of private convex objective functions available to each individual agent. In this context, discrete-time distributed optimization algorithms have been studied extensively (see e.g., [73, 116] and references therein).

There exists another body of literature on distributed continuous-time optimization algorithms (see e.g., [61, 57, 129, 115, 28, 79, 65, 58]). The distributed continuous-time optimization algorithms have applications in coordinated control of multi-agent teams. For example, multiple physical robots modeled by continuous-time dynamics might need to track a team optimal trajectory. Note that most studies in the literature focus on stationary optimization problems in which both the objective functions and constraints do not explicitly depend on time. However, in many applications, the local performance objectives or engineering constraints may evolve in time, reflecting the fact that the optimal solution could be changing over time and create a trajectory (see e.g., [54, 108, 93, 104]), which makes the design and analysis much more complex. Moreover in practical optimization problems, constraints are sometimes inevitable. In this paper, we are interested in the distributed continuous-time algorithms for time-varying constrained optimization problems.

There are just a few works in the literature addressing the distributed continuous-time optimization problem with time-varying objective functions [95, 41, 20, 117, 110, 82, 42]. Specifically, [95, 41] solve the distributed continuous-time time-varying optimization problems with convex set constraints. However, [95] and [41] are limited to solve, respectively, optimization problems with quadratic objective functions and linear programming optimization problems. Moreover, both [95] and [41] can only achieve bounded tracking errors to the optimal solutions. Ref. [117] addresses a Nash equilibrium seeking problem for non-cooperative games where the Nash equilibrium under consideration can be time varying. However, [117] does not consider state constraints in the game problems. Distributed time-varying resource allocation problems are studied in [20, 110], where time-varying objective functions or time-varying loads are considered. However, both [20] and [110] do not consider nonlinear inequality state constraints. Recently, the second-order optimization methods are proven to work well in centralized time-varying optimization problems (see e.g., [93, 104, 27]). However, their use in distributed settings has been prohibited as they require global information of the network to compute the inverse of the global Hessian matrix. Refs. [82, 42] solve the distributed time-varying optimization problems using second-order optimization methods. However, the algorithm in [42] and the consensus-based algorithm in [82] (Section III.B) are limited to the unconstrained problem with local objective functions that have identical Hessians. While the estimator-based algorithm in [82] (Section III.C) can deal with certain objective functions with nonidentical Hessians, it relies on the distributed average tracking techniques [15] and hence poses restrictive assumptions that the time derivatives of the Hessians and the time derivatives of the gradients of the local objective functions exist and be bounded. In addition, because the estimator-based algorithm has to estimate the Hessian inverse of the global objective function, it necessitates the communication of certain virtual variables between neighbors with increased computation costs. While it is possible to convert the constrained optimization problem to an unconstrained one using penalty methods, the resulting penalized objective functions would not have identical Hessians due to the involvement of the nonuniform local constraint functions (even if the original objective functions would), and they might not satisfy the restrictive assumptions mentioned above. As a result, the algorithms in [82, 42] cannot be applied to address the distributed time-varying constrained optimization problem. For distributed time-varying optimization algorithms in discrete-time settings, the readers are referred to [118, 89]. It is worth mentioning that in the literature on discrete-time time-varying optimization algorithms, all the works can only achieve bounded tracking errors, which are usually related to the sampling rate or step size. The continuous-time and discrete-time algorithms serve in different application domains. In this work, we focus on the continuous-time algorithms, which have applications especially in motion coordination.

1.2.2 Distributed Average Tracking

In distributed average tracking, the agents are coupled through the common task that they try to track the average of a set of reference signals, each of which is available to a single agent and is generally time varying; the task should be completed on the basis of local information and local communication among the agents. Recent years have witnessed a growing interest in the study of distributed average tracking, partially due to its broad applications. Distributed average tracking has found applications in distributed sensor fusion [23] and distributed Kalman filtering [75], where the technique has mainly been applied from an estimation perspective. There are also various applications, where distributed average tracking is employed to design control laws for physical agents. Examples include dynamic region-following formation control [17] and distributed convex optimization [82]. It has been recognized that distributed average tracking has its own unique difficulties and faces not only theoretical but also practical challenges, since the tracking objective of distributed average tracking is time varying and unavailable to any agent.

From the estimation perspective, the goal of the distributed average tracking problem is to, in a distributed manner, fuse information or compute common estimates of certain time-varying quantities of interest. A typical example is to estimate and track the averaged position of a moving target by multiple cameras. In this case, the local reference signal is the position data, sensed by each camera, of the moving target. Some distributed average tracking results from the estimation perspective have been presented in [92, 32, 8, 15, 125, 33, 74]. For example, in [92], the authors propose a linear algorithm to achieve distributed average tracking for reference signals with steady states. A proportional algorithm and a proportional-integral (PI) algorithm are proposed in [32] to achieve distributed average tracking with a bounded tracking error under constant or slowly-varying inputs. Based on the nonsmooth sliding mode control theory for nonlinear systems, [15] presents a distributed nonlinear algorithm to achieve accurate distributed average tracking for time-varying reference signals with bounded derivatives. In order to remove the chattering effect caused by the discontinuous signum function, the authors in [125] propose a class of distributed continuous nonlinear algorithms with, respectively, static and adaptive coupling strengths for signals generated by linear dynamics. Different from [125], our note focuses on distributed average tracking over a weight-unbalanced directed graph, which introduces more challenges than its undirected counterpart. Furthermore, in [33], considering the robustness to initial errors, the authors develop a nonlinear distributed average tracking algorithm for arbitrary reference signals with known bounded derivatives.

From the control perspective, some physical agents cooperatively track a desired tracjectory generated by multiple reference signals. For example, the desired trajectory might be the geometric center of multiple leader robots. In this case, the local reference signal is the state of each leader robot. In practice, the physical agents might have more complicated dynamics than single-integrator dynamics. Some researchers have solved the distributed average tracking problem via linear distributed algorithms [49, 83], and some researchers have employed nonlinear distributed algorithms [18, 34, 126, 17, 127]. Both the linear algorithms and the nonlinear algorithms have their features and advantages while with trade-off. For weight-balanced directed graphs, considering single-integrator dynamics, the authors in [49] investigate a continuous algorithm to make agents track the average of the dynamic inputs with a bounded steady-state error. Recently, the authors in [83] propose a linear distributed algorithm with a chain of two integrators for single-integrator dynamics, which can deal with a class of reference signals with steady deviations among the reference signal velocities. However, in the linear algorithms, a common assumption is that the multiple reference signals tend to constant values, and most of the results cannot guarantee accurate tracking. Therefore, to achieve accurate distributed average tracking, the nonlinear algorithm in [15] is further extended in [18] to double-integrator systems for reference signals with bounded accelerations. To address the distributed average tracking problem for physical agents with nonlinear systems, in [34] the authors introduce an exact distributed average tracking algorithm for systems with heterogeneous unknown nonlinear dynamics, where no constraints are imposed on the input reference signals. Furthermore, a distributed algorithm is developed in [126] for agents with nonlinear dynamics to achieve distributed average tracking in finite time. Distributed average tracking algorithms are proposed for agents with general linear dynamics in [17] and for agents with additional Lipschitz-type nonlinear dynamics in [127], where exact distributed average tracking is achieved. However, the trade-off is that the signum function used in some of the above nonlinear algorithms may cause chattering phenomena.

It should be recognized, nevertheless, that the distributed average tracking works alluded to above are all built upon the assumption that the network topology is either undirected or directed but weight-balanced; both cases are highly idealistic and seldom seen in practice. For example, if a camera is used to get the relative positions between agents, due to the limited field of view, it is possible that one agent can sense another agent but not vice versa. In addition, if the agents use communication devices to exchange information with others, the agents might broadcast at different power levels. As a result, the above situations might result in weight-unbalanced directed graphs. Moreover, if the convergence of an algorithm remains unchanged even after removing a few slow communication links or package loss, which might in turn result in a weight-unbalanced directed graph, the algorithm will be more robust and reliable. In order to solve distributed average tracking for generic directed networks, the authors in [6] propose a distributed algorithm to drive the states of all agents to a neighborhood of the average of the reference signals. A prerequisite for the algorithm to work is that the left eigenvector, corresponding to the zero eigenvalue, of the Laplacian matrix should be available to the agents, which is seldom possible in practice, particularly for large networks.

1.2.3 Packet Routing

Modern communication networks have become very challenging mainly due to the following two reasons [11]. Firstly, communication networks have become very complicated and highly dynamic, which makes them hard to model and control. For example, in vehicular and ad hoc networks, nodes frequently move, and link failures might occur during working hours, which might result in topology changes [35, 40]. Second, as the scale of networks continue to multiply, a central controller may be costly to install and slow to configure and be robust to malicious attacks [90, 3]. Therefore, there is a need to develop innovative ways in which traffic routing does not rely on accurate mathematical models and can be managed in a distributed manner. Examples of distributed path planning such as ant colony optimization and swarm approaches have shown success in static environments, but still need more learning for dynamic environments [84].

There are multiple routing algorithms in the literature of traditional packet routing [60, 45, 19]. Among these traditional packet routing algorithms, the shortest path algorithm is the most commonly used routing algorithm [2]. The shortest path algorithm aims to find the shortest path between source and destination nodes and get the packet delivered to the destination node as quickly as possible. The shortest path algorithm is regarded as the best routing algorithm on lower network load since packets can be delivered using the least amount of time along the shortest path between two nodes provided that there is no congestion along the route. However, when the network load is high, the shortest path algorithm will cause a serious backlog in busy routers. Another problem with the shortest path algorithm is that it relies on having full knowledge of the network topology to design routing algorithms and hence needs manual adjustment when topology or traffic changes happen.

Using reinforcement learning for packet routing has attracted increasing interest recently. Various reinforcement learning methods have been proposed to deal with this classical communication network problem and achieved better performances compared with traditional routing methods. Applications of traditional RL to solve packet routing problems started in the early 1990s with the seminal work [12], where Q-routing was proposed. Q-routing [12] is an adaptive routing approach based on a reinforcement learning algorithm known as Q-learning. Q-routing routes packets based on the learned delivery times (Q values) and achieves a much smaller average delivery time compared with the benchmark shortest-path algorithm [103]. Since then, several extensions of Q-routing have been proposed, e.g., Dual Q-routing [52], Predictive Q-routing [21], Full Echo Q-routing [46], Hierarchical Q-routing [62] and Ant-based Q-routing [94]. However, Q-routing is a value-based RL algorithm. That is, Q-routing is a deterministic algorithm that might cause traffic congestion at high loads and does not distribute incoming traffic across the available links. Due to the drawbacks of a value-based algorithm, some researchers begin to consider policy-based RL algorithms for packet routing problems [105, 77]. Since policy-based RL algorithms can explore the class of stochastic policies, it is natural to expect policy-based algorithms to be superior for certain types of network topologies and loads, where the optimal policy is stochastic. In [77], the results show that policy-based RL algorithms perform better than value-based algorithms, especially on high flow load. These traditional reinforcement learning routing algorithms use tabular functions or simple algebraic functions to estimate the Q functions or policy functions. This is limiting for a large number of states and thus cannot take full advantage of the network traffic history and dynamics. In this work, we investigate the policy-based deep reinforcement learning algorithm and use deep neural networks to approximate the policy function. The combination of deep learning techniques with reinforcement learning methods can learn useful representations for the routing problems with high dimensional raw data input and thus achieve superior performance.

Deep reinforcement learning is the combination of reinforcement learning and deep learning, which has been able to solve a wide range of complex decision-making tasks. However, rare works are investigating how deep reinforcement learning can be leveraged for packet routing problems since the wireless network is a multi-agent environment and the network environment is non-stationary from the perspective of any individual router. This prevents the straightforward use of experience replay, which is crucial for stabilizing deep Q learning [64]. [71] combines the Q-routing and deep Q-learning to solve the routing problem. However, the training process of the algorithm proposed in [71] is in a centralized manner (all the routers need to share parameters), which might cause issues in real-world large-scale network environments. The authors in [114] propose to use a deep actor-critic reinforcement learning algorithm to optimize the performance of the communication network. However, the training and testing process in [114] are also in a centralized manner. Recently, distributed Deep Q-routing has been proposed in [119], where deep recurrent neural network (LSTM) has been utilized to tackle the non-stationary problem in multi-agent reinforcement learning. However, in [119], it is assumed that the bandwidth of each link equals the packet size, in which case only a single packet can be transmitted at a time.

There is a huge body of literature on single-agent deep reinforcement learning algorithms, where the environment stays largely stationary. Unfortunately, traditional deep reinforcement learning algorithms are poorly suited to multi-agent environments, where the environment becomes non-stationary from the perspective of any individual agent. This might cause divergence for value-based reinforcement learning and very high variance for policy-based reinforcement learning algorithms. In the literature, researchers propose multiple methods to apply reinforcement learning algorithms in multi-agent settings. To name a few, centralized training and distributed execution [64, 43, 55], distributed training and execution under fully-observable environments [121], and independent Q-learning [37, 67, 31]. However, independent Q-learning is a value-based reinforcement learning algorithm, and in this work, we aim to investigate the policy-based reinforcement learning routing algorithm. The ideas of centralized training and fully observable state space work well when there exists a small number of agents in the communication network. With increasing the number of agents, the volume of the information might overwhelm the capacity of a single unit. To tackle this problem, one effective idea to remove the central unit and only allow the agents to share information with only a subset of agents, to reach a consensus over a variable with these agents (called neighbors) [109, 120].

1.3 Contributions

In this dissertation, we focus on the following two problems for multi-agent systems:

1. Distributed convex optimization with time-varying objective functions.

2. Packet routing problem in a network environment in the presence of link failures.

Some materials of this dissertation have been published in [100, 96, 98, 102, 99]. In this section, we briefly talk about the contributions of this dissertation as follows.

- The first part of this dissertation aims to develop distributed algorithms to solve the continuous-time optimization problems with private time-varying convex objective functions for undirected topologies. We propose multiple optimization algorithms for different application scenarios. The main contributions are given as follows.

  - First, we propose a distributed nonsmooth algorithm with state-dependent gains for the unconstrained case. Here the interaction gain of each agent is adjusted according to the variation of the Hessian and gradient information of the convex local objective functions, so that the algorithm can solve the time-varying optimization problem without imposing a bound on any information about the local objective functions. Therefore, the proposed algorithm can deal with more general objective functions. To the best of our knowledge, this is the first work in the literature of distributed continuous-time time-varying optimization that the optimization problem can be solved without imposing a bound on any information about the local objective functions. It is shown that all agents achieve consensus in finite time and the consensus solution converges to the optimal solution asymptotically. Numerical simulations are presented to illustrate the theoretical results. Moreover, the proposed algorithm is experimentally validated on a multi-Crazyflie platform.

– Second, for the case where there exist common time-varying linear equality constraints, an extended algorithm is proposed. Local Lagrangian functions are introduced to address the equality constraints. Similarly, the time-varying constrained optimization problem can be solved without imposing a bound on any information about the local objective functions and constraint functions. Note that the distributed time-varying constrained optimization problem has its unique difficulties and is more challenging than the unconstrained counterpart since the local constraints are also time varying. Numerical simulations are presented to illustrate the theoretical results.

– For the case where there exist only time-varying inequality constraints, we develop a sliding-mode method with a Hessian-dependent gain for all the agents to achieve consensus on the states. Meanwhile, a Hessian-based (second-order) optimization method coupled with the log-barrier penalty functions is proposed to track the local time-varying optimal solution. Although [111, 27] also use log-barrier penalty functions to address the inequality constraints, our work is the first to leverage the log-barrier penalty functions to the distributed time-varying optimization problems. To implement the algorithm, each agent just needs its own state and the relative states between itself and its neighbors. When the agents' states are their positions, the algorithm can be implemented based on purely local sensing (e.g., absolute and relative positions) without the need for communicating virtual variables. The asymptotical convergence to the optimal solution is established based on nonsmooth analysis, Lyapunov theory and convex optimization theory. Both numerical simulation and real experimental results are presented to illustrate the effectiveness of the theoretical results. To the best of our knowledge, this is the first work in the literature on distributed continuous-time optimization with time-varying inequality constraints that guarantees zero tracking errors.

  - Furthermore, we extend the previous result with the following improvements. We add quadratic penalty functions to account for equality constraints to make the algorithm be applicable to more general problems and we present an adaptive control gain design under which the restriction on knowing the upper bounds on certain prior information is removed. And the asymptotical convergence of the extended algorithm to the vicinity of the optimal solution is studied under suitable assumptions. Numerical simulation results are presented to illustrate the effectiveness of the theoretical results.

- The second part of the dissertation is devoted to studying distributed time-varying optimization for generic directed networks, which are possibly weight-unbalanced. We aim to investigate distributed time-varying optimization with quadratic objective functions, which is equivalent to distributed average tracking problems. To the best of our knowledge, the distributed average tracking problem has not yet been addressed in the literature for weight-unbalanced directed graphs without knowing the left eigenvector of the Laplacian matrix. Specifically, we introduce two algorithms for different application scenarios, each of which has its own relative benefits. The main contributions are given as follows.

- In the first algorithm, we consider single-integrator dynamics and avoid the use of the left eigenvector of the Laplacian matrix. The proposed algorithm accounts for a generic directed network and a wide class of time-varying reference signals of which the accelerations have bounded deviations; hence, it is practically more relevant and meaningful. Particularly, we introduce a distributed linear algorithm with a chain of two integrators coupled with a distributed estimator for the left eigenvector of the Laplacian matrix associated with the zero eigenvalue. We prove that if the deviations among the reference signal accelerations tend to zero (respectively, bounded), the algorithm can achieve distributed average tracking with zero (respectively, bounded) tracking error.
- In the second algorithm, we consider agents with high-order integrator dynamics. We propose a distributed nonlinear algorithm coupled with a distributed estimator for the left eigenvector of the Laplacian matrix associated with the zero eigenvalue. In addition, we replace the signum function in [17] with a continuous approximation in order to remove the chattering effect caused by the discontinuous signum function. The approximate function is widely adopted in the sliding mode control field [26]. The results show that if the reference signals and signal control inputs are bounded, the algorithm can achieve distributed average tracking with arbitrarily small tracking errors. The convergence of the algorithm to

the vicinity of the average of the reference signals is established via Lyapunov stability theory and input-to-state stability theory.

- The last part of this dissertation investigates the challenge of how can one build an adaptive network routing controller that continue to provide optimum network performance, even when the topology changes. For this we model the challenge as a path optimization technique that can be adaptive to various network load and topology changes, via novel deep reinforcement learning. Modern communication networks are highly dynamic and hard to model and predict, therefore, we aim to develop a novel *experience-driven* algorithm that can learn to select paths from its experience rather than an accurate mathematical model. Additionally, due to difficulties in installing a central node to gather information from large-scale and widely distributed routers, we design a distributed optimization framework to learn the local optimal strategy. Our specific contributions are:

  - Using a policy-based deep reinforcement learning method, we train the model at a variety of network loads and save the optimal neural network. Once deployed, our trained neural network can perform superiorly at high network load compared to value-based RL learning.

  - Our neural networks are optimized for multi-objective optimization for both packet delivery time and packet loss on the network links. We design an appropriate reward (utility) function, which well represents the preference of the network controller, to minimize both packet delivery time and packet loss when link failures occur.

- Our proposed MAMRL framework aims to make good online decisions under the guidance of powerful Deep Neural Networks (DNNs). In addition, by leveraging the model-agnostic meta-learning technique [30], our neural networks can quickly alternate paths to minimize both packet delivery time and packet loss when link failures occur.

- Our neural network model is deployed per multiple agents to represent multiple routers, allowing each router agent to learn and optimize the traffic routing based on their local information. To achieve this, we leverage a dynamic consensus estimator [50] to diffuse local information and estimate global rewards, still achieving the best average packet delivery time.

1.4 Organization

This dissertation is organized as follows. In Chapter 2, we describe some preliminaries on graph theory, nonsmooth analysis, convex optimization and reinforcement learning. In Chapter 3, we consider distributed time-varying optimization problems for undirected topologies and show four algorithms for different application scenarios. In Chapter 4, we focus on distributed average tracking for directed graphs. We propose two algorithms, respectively, for single-integrator and high-order integrator dynamics. In Chapter 5, we investigate policy optimization algorithm for packet routing problems in the presence of link failures. Finally, in Chapter 6, we give a summary of this dissertation.

Chapter 2

Preliminaries

In this Chapter, we introduce some preliminaries used in this dissertation.

2.1 Notation

The following notations are defined throughout this dissertation to avoid possible ambiguities. Let $\mathbb{R}, \mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ denote the sets of real numbers, real vectors of dimension n, and real matrices of size $n \times m$, respectively. Let $\mathbb{R}_{>0}$ represent the set of positive real numbers. Let $\mathbf{1}_n$ (resp. $\mathbf{0}_n$) be the vector of n ones (resp. n zeros), I_n denote the $n \times n$ identity matrix, and $\bar{\mathbf{0}}_n$ (resp. $\bar{\mathbf{0}}_{m\times n}$) denote the $n\times n$ (resp. $m\times n$) matrix of all zeros. For a matrix $A \in \mathbb{R}^{m\times n}$, $\sigma_{\max}(A)$ denotes the maximal singular value of matrix A, A^T is the transpose of A, and $\text{vec}(A) = [\text{col}_1(A)^T, \cdots, \text{col}_n(A)^T]^T \in \mathbb{R}^{nm}$ is the column vector of size $nm\times 1$ obtained by stacking the columns of A, where $\mathrm{col_i(A) \in \mathbb{R}^m}$ represents the ith column of matrix A. For a square matrix $A \in \mathbb{R}^{m\times m}$, A^{-1} denotes the inverse of A. For a vector $x \in \mathbb{R}^{n\times 1}$, $\text{diag}(x) \in \mathbb{R}^{n\times n}$ represents the diagonal matrix with the elements in the main diagonal being the elements of x, $\|x\|_p$ denotes the p-norm of the vector x, $B(x, \delta)$ represents the open ball of radius δ centered at x, and $\mathrm{sgn}(x) = [\mathrm{sgn}(x_1), \cdots, \mathrm{sgn}(x_n)]^T$, where $\mathrm{sgn}(x_i) = -1$ if $x_i < 0$, $\mathrm{sgn}(x_i) = 1$ if $x_i > 0$, and otherwise $\mathrm{sgn}(x_i) = 0$. Let $\nabla f(x,t)$ and $\nabla^2 f(x,t)$ denote, respectively, the gradient and Hessian of the function $f(x,t) : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \mapsto \mathbb{R}$ with respect to the vector x. Let $\otimes$ be the Kronecker product. Let $n!$ be the product of n consecutive natural numbers from 1 to n. Let $f_2 \circ f_1(\cdot)$ be the composition of two functions $f_1(\cdot)$ and $f_2(\cdot)$. Let $f^{-1}(\cdot)$ denote the inverse of a function $f(\cdot)$. Let $\lfloor a \rfloor$ be the largest integer that is smaller than or equal to a.

Proposition 1. *[9] Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times l}$ and $D \in \mathbb{R}^{l \times k}$. Then,* $\mathrm{vec}(ABD) = (D^T \otimes A)\mathrm{vec}(B)$.

2.2 Graph Theory

We present some basic definitions and relations regarding graphs in this section.

Graph

A graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where

- $\mathcal{V} = \{1, ..., n\}$ is the node set,
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set,
- $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$ is the weighted adjacency matrix with $a_{ij} \in \mathbb{R}_{>0}$ if $(j, i) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise.

An edge (j, i) implies that node i can receive information from j. Let $\mathcal{N}_i = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\}$ denote the set of in-neighbors of node i. A path is a sequence of nodes connected by edges.

The Laplacian matrix $\mathcal{L} = [l_{ij}] \in \mathbb{R}^{n\times n}$ associated with $\mathcal{A}$ is defined as $l_{ii} = \sum_{j=1,j\neq i}^{n} a_{ij}$ and $l_{ij} = -a_{ij}$, where $i \neq j$. Note that $\mathcal{L}\mathbf{1}_n = \mathbf{0}_n$.

Undirected graph and directed graph

An undirected graph is a graph where all the edges are bidirectional. In contrast, a graph where there exists at least one edge point in a direction is called a directed graph. Moreover, $a_{ij} = a_{ji}$ for all $i, j \in \mathcal{V}$ for undirected graphs, yet it is possible that $a_{ij} \neq a_{ji}$ for directed graphs

Unbalanced graph and balanced graph

All the undirected graphs are balanced graphs. A directed graph is balanced if and only if $\mathbf{1}_n^T\mathcal{L} = \mathbf{0}_n^T$.

Connected graph and stronlgy connected graph

An undirected graph is connected if for every pair of nodes there is a path connecting them. A directed graph is strongly connected if for every pair of nodes there is a directed path connecting them.

Lemma 2. *[76] [56] Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ be a directed graph with the Laplacian matrix $\mathcal{L} \in \mathbb{R}^{n\times n}$. If $\mathcal{G}$ is strongly connected, then the following statements hold.*

- *There exists a positive left eigenvector $p = [p_1, ..., p_n]^T$ of $\mathcal{L}$ associated with the zero eigenvalue, such that $p_i > 0, i = 1, ..., n$, $p^T\mathcal{L} = \mathbf{0}_n^T$, and $\sum_{i=1}^{n} p_i = 1$.*

- *The Laplacian matrix $\mathcal{L}$ has a simple zero eigenvalue corresponding to the right eigenvector $\mathbf{1}_n$, and all the nonzero eigenvalues have positive real parts.*

- $\min_{a^T x=0, x\neq \mathbf{0}_n} x^T\bar{\mathcal{L}}x > \lambda_2(\bar{\mathcal{L}})x^Tx/n$, *where x is any vector, $\bar{\mathcal{L}} = \mathcal{L}^TP + P\mathcal{L}$ and $P = \mathrm{diag}(p)$, a is any vector with positive entries, and $\lambda_2(\bar{\mathcal{L}})$ is the smallest nonzero* *eigenvalue of matrix* $\bar{\mathcal{L}}$.

- $\lim_{t\to\infty} \exp(-\mathcal{L}t) = \mathbf{1}_n p^T$.

2.3 Nonsmooth Analysis

In this section, we recall some important definitions of the nonsmooth systems that will be exploited in our main result.

Definition 3. *(Filippov Solution)[87] Consider the vector differential equation*

$$\dot{x} = f(x,t), \tag{2.1}$$

where $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ *is Lebesgue measurable and locally essentially bounded. A vector function* $x(\cdot)$ *is called a Filippov solution of (2.1) on* $[t_0, t_1]$*, if* $x(\cdot)$ *is absolutely continuous on* $[t_0, t_1]$ *and for almost all* $t \in [t_0, t_1]$*,* $\dot{x}(t) \in K[f](x,t)$*, where* $K[f](x,t) := \bigcap_{\delta>0} \bigcap_{\mu(N)=0} \overline{\text{co}} f\big(B(x,\delta) - N, t\big)$ *is the Filippov set-valued map of* $f(x,t)$ *and* $\bigcap_{\mu(N)=0}$ *denotes the intersection over all sets* N *of Lebesgue measure zero.*

Definition 4. *(Clarke's Generalized Gradient) [87] Consider a locally Lipschitz continuous function* $V(x) : \mathbb{R}^d \to \mathbb{R}$*, the generalized gradient of the function* V *at* x *is given by* $\partial V(x) := \overline{\text{co}}\{\lim \nabla V(x_i) | x_i \to x, x_i \notin \Omega_V\}$*, where* Ω_V *is the set of Lebesgue measure zero where the gradient of* V *is not defined.*

Definition 5. *(Chain Rule)[87] Let* $x(\cdot)$ *be a Filippov solution of* $\dot{x} = f(x,t)$ *and* $V(x) : \mathbb{R}^d \to \mathbb{R}$ *be a locally Lipschitz continuous function. Then for almost all* t*,*

$$\frac{d}{dt} V[x(t)] \in \dot{\tilde{V}},$$

where $\dot{\tilde{V}}$ *is the set-valued Lie derivative defined as* $\dot{\tilde{V}} := \bigcap_{\xi\in\partial V} \xi^T K[f]$.

2.4 Convex Optimization

In this section, we review some basics in convex optimization. Specifically, we list some standard definitions and properties in convex optimization. The details can be found in Ref. [13].

Convex functions: A function $f(x) : \mathbb{R}^p \to \mathbb{R}$ is convex if, for any points $x, y \in \mathbb{R}^p$, and $\theta \in [0, 1]$, is satisfies that

$$f[\theta x + (1-\theta)y] \leq \theta f(x) + (1-\theta) f(y). \tag{2.2}$$

Gradients and Hessian: The gradient of a function $f(x) : \mathbb{R}^p \to \mathbb{R}$ (assume it is differentiable), $\nabla f(x)$, is the vector-valued function whose values at a point x are the partial derivatives of $f(x)$ with respect to x:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_p}(x) \end{bmatrix}, \tag{2.3}$$

where x_i is the ith element of vector x.

The Hessian of a function $f(x) : \mathbb{R}^p \to \mathbb{R}$ (assume it is twice differentiable), $\nabla^2 f(x)$, is a $p \times p$ square matrix, which is defined as follows:

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 x_p}(x) \\ \frac{\partial^2 f}{\partial x_2 x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_2 x_p}(x) \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial x_p x_1}(x) & \frac{\partial^2 f}{\partial x_p x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_p^2}(x) \end{bmatrix}. \tag{2.4}$$

Strong convexity: A convex function $f(x) : \mathbb{R}^p \to \mathbb{R}$ is further said to be strongly convex if it is twice differentiable and there exists a positive scalar m such that for any points x, y, it satisfies

$$[\nabla f(x) - \nabla f(y)]^T (x - y) \geq m\|x - y\|^2. \tag{2.5}$$

2.5 Reinforcement Learning

In this section, we introduce some important definitions of reinforcement learning that will be used in this dissertation. Reinforcement learning is concerned with how an intelligent agent learns a good strategy from experimental trials and relative feedback received. With the optimal strategy, the agent is capable to actively adapt to the environment to maximize cumulative rewards. Almost all the deep RL problems can be framed as Markov Decision Processes (MDPs), which consists of four key elements $\langle \mathcal{S}, \mathcal{A}, P, R \rangle$. More specifically, at each decision epoch t, the intelligent agent can stay in state s_t that belongs to the state space $\mathcal{S}$ of the environment, and choose to take an action a_t that belongs to the action space $\mathcal{A}$ to switches from one state to another. The probability that the process moves into its new state s_{t+1} is given by the state transition function $P(s_{t+1}|s_t, a_t)$. Once an action is taken, the environment delivers a reward r as feedback. Figure 2.1 shows the general process of reinforcement learning (the definition of policy will be given below).

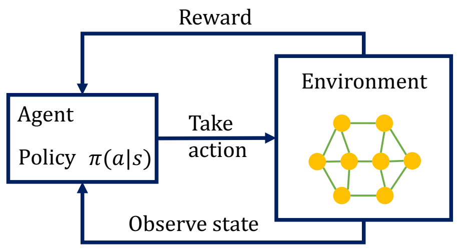

Figure 2.1: A global reinforcement learning agent learning network states.

There are two key functions in RL: **policy function** $\pi(a_t|s_t)$ and **value function** $V_\pi(s_t)/Q_\pi(s_t, a_t)$. The policy is a mapping from state s_t to action a_t and tells the agent which action a_t to take in state s_t. For example, in the path optimizing problem, the policy is the router's strategy that finds the best adjacent router to send out current packets given the current utilization state of the communication network. The state value function $V_\pi(s_t)$ measures how rewarding a state is under policy π by a prediction of future reward. Similarly, the action-state value function $Q_\pi(s_t, a_t)$ tells, for a given policy, what the expected cumulative reward of taking action a_t in state s_t is.

The goal of RL is to find the optimal policy that achieves optimal value functions: $\pi^* = \text{argmax}_\pi V_\pi(s_t) = \text{argmax}_\pi Q_\pi(s_t, a_t)$. Traffic engineering is a natural application of RL by exploring with different routing policies, gathering statistics about which policies maximize the utility function, and learning the best policy accordingly.

Value-based algorithms versus Policy-based algorithms: Value-based RL algorithms attempt to learn the tabular or approximation of the state-action value $Q_\pi(s, a)$ and selects the action based on the maximal value function of all available actions for a given state. For example, the Q-routing algorithm enables the routers to restore Q-values as the estimate of the negative transmission time between that router and the others. To shorten the average packet delivery time, routers will choose the action with maximal Q-values. Policy-based RL algorithms instead learn the policy directly with a parameterized function concerning θ, $\pi_\theta(a_t|s_t)$ and train the policy to maximize the expected cumulative reward function. Policy-based algorithms can learn stochastic policies. It is worthwhile to note that stochastic means stochastic in some action-state pairs where it makes sense. Usually, value-based algorithms, which choose the actions with the maximal values, can only follow deterministic policies or stochastic policies with predetermined distributions. That is not quite the same as learning the real optimal stochastic policy. Since the current communication networks are highly dynamic and stochastic, we can expect that the policy-based RL algorithms perform superiorly to the value-based RL algorithms for certain scenarios, where the optimal policy is stochastic. In this work, to enable high-dimensional state representations (such as action histories), we consider deep RL algorithms, which adopt deep neural networks to approximate the policy functions. Here the policy parameters θ are the weights of the deep neural networks.

Chapter 3

Distributed Convex Optimization with Time-Varying Objective Functions

In this Chapter, we address distributed continuous-time optimization problems with time-varying objective functions for undirected topologies. The goal is for multiple agents to cooperatively minimize the sum of local time-varying objective functions with only local interaction and information. Here, the optimal point is time varying and creates an optimal trajectory. First, for the unconstrained case, a distributed nonsmooth algorithm coupled with a state-dependent gain is proposed. It is shown that the interaction gain for each agent can be computed according to the variation of the Hessian and gradient information of the convex local objective functions, so that the algorithm can solve the time-varying optimization problem without imposing a bound on any information about the local objective functions. Second, for the case where there exist common time-varying linear equality constraints, an extended algorithm is presented, where local Lagrangian functions are introduced to address the equality constraints. Third, for the case where there exist only time-varying nonlinear inequality constraints, we present a distributed control algorithm that consists of a sliding-mode consensus part and a Hessian-based optimization part coupled with the log-barrier penalty functions. The algorithm can guarantee the asymptotical tracking of the optimal solution with a zero tracking error. Finally, we extend the previous result to the case where there exist not only time-varying nonlinear inequality constraints but also linear equality constraints. An extended algorithm is presented, where quadratic penalty functions are introduced to account for the equality constraints and an adaptive control gain is designed to remove the restriction on knowing the upper bounds on certain information. The asymptotical convergence of the extended algorithm to the vicinity of the optimal solution is studied under suitable assumptions. The effectiveness of the proposed algorithms is illustrated in simulation. In addition, two proposed algorithms are applied to a multi-robot navigation problem with experimental demonstration on a multi-Crazyflie platform to validate the theoretical results.

3.1 Distributed Time-Varying Optimization With State-Dependent Gains

3.1.1 Problem Formulation

Consider a multi-agent system consisting of n agents with an interaction topology described by the undirected graph $\mathcal{G}(t)$. Each agent can interact only with its local neighbors. Suppose that the agents satisfy the following single-integrator dynamics

$$\dot{x}_i(t) = u_i(t), \tag{3.1}$$

where $x_i(t) \in \mathbb{R}^m$ and $u_i(t) \in \mathbb{R}^m$ are the state and control input of agent i. Our goal here is to design $u_i(t)$ using only local information and interactions with neighbors, such that all the agents cooperatively find the optimal solution $y^*(t) \in \mathbb{R}^m$ (assuming it exists for all $t \geq 0$) which is defined as

$$y^*(t) = \underset{y(t)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} f_i[y(t), t] \right\}, \tag{3.2}$$

where $f_i[y(t), t] : \mathbb{R}^m \times \mathbb{R}_{\geq 0} \mapsto \mathbb{R}$ are the local objective functions. It is assumed that $f_i[y(t), t]$ is only known to agent i and is twice continuously differentiable with respect to $y(t)$ and continuously differentiable with respect to t. Note that $\sum_{i=1}^{n} f_i[x_i(t), t] = \sum_{i=1}^{n} f_i[y(t), t]$, if $x_i(t) = x_j(t) = y(t)$ for all $i, j \in \mathcal{V}$, the above problem (3.2) is equivalent to finding the optimal solution $x^*(t) \in \mathbb{R}^{m*n}$ which is defined as

$$\begin{aligned} x^*(t) \;&=\; \underset{x(t)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} f_i[x_i(t), t] \right\}, \\ \text{s.t.} \qquad & x_i(t) = x_j(t), \quad \forall i, j \in \mathcal{V}, \end{aligned} \tag{3.3}$$

where $x(t)$ is the vector that concatenates the state vectors $x_i(t) \in \mathbb{R}^m$ of all the agents. Note that problem (3.2) will be solved as a consensus minimization problem with the time-varying team objective function $\sum\limits_{i=1}^{n} f_i[x_i(t), t]$. Here, the goal is that each state $x_i(t)$ converges to the optimal solution $y^*(t)$, i.e.,

$$\lim_{t \to \infty} [x_i(t) - y^*(t)] = \mathbf{0}_m. \tag{3.4}$$

We introduce the following assumptions and the following lemma which are all standard in the recent literature [82].

Assumption 1. *The graph $\mathcal{G}(t)$ is undirected and connected for all $t \geq 0$.*

Assumption 2. *The length of the time interval between any two contiguous switching topologies is greater than or equal to a given positive constant.*

Arbitrary switching of the graph $\mathcal{G}(t)$ might lead to the Zeno behavior. Hence Assumption 2 is imposed to prevent the system from exhibiting the Zeno behavior.

Assumption 3. *Each objective function $f_i[x_i(t), t]$ is uniformly strongly convex in $x_i(t)$ and its Hessian matrix $\nabla^2 f_i[x_i(t), t]$ is identical under identical local states $x_i(t)$ for all $t \geq 0$, i.e., $\nabla^2 f_i[x_i(t), t] \geq \alpha I_m$, for some $\alpha > 0$ and $\nabla^2 f_i[x_i(t), t] = \nabla^2 f_j[x_j(t), t]$ if $x_i(t) = x_j(t)$ for all $i, j \in \mathcal{V}$.*

The uniform strong convexity of each objective function $f_i[y(t), t]$ implies the uniform strong convexity of $\sum_{i=1}^{n} f_i[y(t), t]$, such that the optimal trajectory $y^*(t)$ (assuming it exists for all $t \geq 0$) is unique for all $t \geq 0$. Moreover, due to the equivalence between (3.2) and (3.3), the optimal solution $x^*(t)$ defined in (3.3) is unique for all $t \geq 0$.

Remark 6. *Assumption 3 requires that the Hessian matrix $\nabla^2 f_i[x_i(t), t]$ be identical under identical local states $x_i(t)$ for all $t \geq 0$, which might be restrictive. However, in the literature of distributed continuous-time time-varying optimization, it is common to assume that all the Hessian matrices $\nabla^2 f_i[x_i(t), t]$ are identical, i.e., $\nabla^2 f_i[x_i(t), t] = \nabla^2 f_j[x_j(t), t]$ for all $i, j \in \mathcal{V}$ and all $t \geq 0$ (see [82], [42]). In this paper, we reconsider the identical Hessian assumption, and relax it further. The algorithms herein do not need $\nabla^2 f_i[x_i(t), t] = \nabla^2 f_j[x_j(t), t]$ for all $x_i(t)$ and $x_j(t)$. Instead, they only need $\nabla^2 f_i[x_i(t), t] = \nabla^2 f_j[x_j(t), t]$ when $x_i(t) = x_j(t)$. Note that the identical Hessian condition can be satisfied in many sit-*

uations, e.g., $f_i[x_i(t),t] = [\alpha\, x_i(t) + b_i(t)]^2$ with a positive constant α and a time-varying function $b_i(t)$, which is commonly used for robot control and energy minimization.

Lemma 7. *[13] Let $f(r) : \mathbb{R}^m \to \mathbb{R}$ be a continuously differentiable convex function with respect to r. The function $f(r)$ is minimized at r^* if and only if $\nabla f(r^*) = 0$.*

3.1.2 Algorithm Design

This subsection presents and analyzes a distributed adaptive control algorithm for the time-varying optimization problem in (3.3). The controller for agent i is:

$$\begin{aligned} u_i(t) &= \phi_i(t) - \sum_{j\in\mathcal{N}_i(t)} \Big\{[\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j]\mathrm{sgn}[x_i(t) - x_j(t)]\Big\}, \\ \phi_i(t) &= -\left\{\nabla^2 f_i[x_i(t),t]\right\}^{-1} \left\{\nabla f_i[x_i(t),t] + \frac{\partial}{\partial t}\nabla f_i[x_i(t),t]\right\}, \end{aligned} \tag{3.5}$$

where $\gamma_i \in \mathbb{R}_{>0}$ is a constant control gain. The auxiliary variables $\phi_i(t)$ and $\phi_j(t) \in \mathbb{R}^m$ automatically adjust the gain of the interaction term $\mathrm{sgn}[x_i(t) - x_j(t)]$ for $j \in \mathcal{N}_i(t)$.

Remark 8. *Algorithm* (3.5) *solves the time-varying optimization problem of* (3.2) *as a consensus minimization problem with the time-varying team objective function $\sum_{i=1}^{n} f_i[x_i(t),t]$. The term $-\sum_{j\in\mathcal{N}_i(t)}\Big\{[\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j]\ \mathrm{sgn}[x_i(t) - x_j(t)]\Big\}$ is introduced to achieve consensus among the agents, that is $x_i(t) \to x_j(t)$, $\forall i,j \in \mathcal{V}$. The auxiliary variable $\phi_i(t)$ is employed to force the consensus state to track the optimal solution $y^*(t)$. Note that the three terms in $\phi_i(t)$, namely, $\{\nabla^2 f_i[x_i(t),t]\}^{-1}$, $\nabla f_i[x_i(t),t]$ and $\partial\nabla f_i[x_i(t),t]/\partial t$, could become unbounded due to the involvement of $x_i(t)$ and t. Here, the state-dependent gain $\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j$ is used to overcome the possible unboundedness of $\phi_i(t)$.*

Remark 9. *Algorithm* (3.5) *is distributed, because each agent only uses information about its own objective function and information communicated by its neighbors. Take agent i*

as an example. Agent i uses its own information: $x_i(t)$ and the Hessian and gradient information of its objective function $f_i[x_i(t), t]$; as well as information received from its neighbors: $x_j(t) \in \mathbb{R}^m$, $\gamma_j \in \mathbb{R}_{>0}$ and $\|\phi_j(t)\|_\infty \in \mathbb{R}$ for $j \in \mathcal{N}_i(t)$. Moreover, in order to implement the proposed algorithm (3.5)*, all agents need to share a common coordinate system. Note that many global coordinate systems exist worldwide, such as GPS.*

Remark 10. *The design of the algorithm* (3.5) *is partly motivated by the algorithm given in [82]:*

$$\begin{aligned} u_i(t) &= -\sum_{j\in\mathcal{N}_i} \beta_{ij}(t)\ \mathrm{sgn}[x_i(t) - x_j(t)] + \phi_i(t), \\ \dot{\beta}_{ij}(t) &= \|x_i(t) - x_j(t)\|_1,\ j \in \mathcal{N}_i, \end{aligned} \tag{3.6}$$

where $\phi_i(t)$ is the same as that in (3.5)*. Compared with the algorithm* (3.6)*, the algorithm* (3.5) *has two advantages. First, the algorithm in [82] places a bound on the Hessians and the rate of the change of the gradients of the local objective functions. These requirements can limit the applicable class of objective functions. For example, as stated in Remark 6, a commonly used objetive function for robot control and energy minimization is $f_i[x_i(t), t] = [\alpha x_i(t) + b_i(t)]^2$. If $\dot{b}_i(t)$ is unbounded, then it is obvious that it does not satisfy the above requirement. In this paper, we introduce a novel state-dependent control gain design to remove the above requirement. The proposed algorithm* (3.5) *can deal with more general objective functions that cannot be handled by [82]. Second, the adaptive control gain $\beta_{ij}(t)$ designed in* (3.6) *keeps increasing until the consensus is achieved. Therefore, the control gain might become unnecessarily large. In contrast, the state-dependent gain in this paper is computed such that the above situation is eliminated. The state-dependent control gain approach introduces new theoretical challenges that are the focus of this paper. However, the* *algorithm* (3.5) *has a disadvantage as well. That is, each agent is required to be able to get the information of the variable* $\phi_j(t)$ *and* γ_j *from its neighbors, which requires the existence of the communication capabilities, while the algorithm* (3.6) *can be implemented using only local sensing without the need for the existence of the communication capabilities as long as the relative position* $(x_i - x_j)$ *between each agent and its neighbors can be measured.*

3.1.3 Algorithm Analysis

This subsection establishes the asymptotic convergence of system (3.1) under controller (3.5) to the optimal solution in (3.2).

Lemma 11. *Given Assumptions 1 and 2, using* (3.5) *for* (3.1), *all the states* $x_i(t)$ *will achieve consensus in finite time, i.e., there exists a time* T *such that* $\|x_i(t) - x_j(t)\|_2 = 0$ *for all* $i, j \in \mathcal{V}$ *and for all* $t > T$.

Proof. The main idea of our proof is to show that each corresponding component of the agents' state vectors reaches a consensus separately in finite time. Let $\dot{x}_{ik}(t)$, $x_{ik}(t)$ and $\phi_{ik}(t)$ denote, respectively, the kth components of $\dot{x}_i(t)$, $x_i(t)$ and $\phi_i(t)$. Define

$$A_{1k}(t) \triangleq \{i \mid x_{ik}(t) = \max_{i \in \mathcal{V}}[x_{ik}(t)]\},$$

$$A_{2k}(t) \triangleq \{i \mid x_{ik}(t) = \min_{i \in \mathcal{V}}[x_{ik}(t)]\},$$

$$\bar{x}_k(t) \triangleq \frac{1}{|A_{1k}(t)|} \sum_{i \in A_{1k}(t)} x_{ik}(t),$$

$$\underline{x}_k(t) \triangleq \frac{1}{|A_{2k}(t)|} \sum_{i \in A_{2k}(t)} x_{ik}(t),$$

where $|A_{1k}(t)| \geq 1$ and $|A_{2k}(t)| \geq 1$ denote, respectively, the cardinality of $A_{1k}(t)$ and

$A_{2k}(t)$. Clearly, using (3.5) for (3.1), the kth component of each $\dot{x}_i(t)$ can be written as

$$\dot{x}_{ik}(t) = \phi_{ik}(t) - \sum_{j\in\mathcal{N}_i(t)} \Big\{[\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j]\mathrm{sgn}[x_{ik}(t) - x_{jk}(t)]\Big\}. \tag{3.7}$$

We first show that, when $\bar{x}_k(t) \neq \underline{x}_k(t)$, $x_{ik}(t)$ for all $i \in A_{1k}(t)$ are nonincreasing and $x_{ik}(t)$ for all $i \in A_{2k}(t)$ are nondecreasing. Note that even though all elements $x_{ik}(t),\ i \in A_{1k}(t)$ have the same value, they need not have the same derivative. The proof will proceed by induction to establish a contradiction.

Assume that for agent $l \in A_{1k}(t)$ there exist two time instants $t_1 < t_2$ such that $\dot{x}_{lk}(t) > 0$ for all $t \in [t_1, t_2]$ almost everywhere (i.e., except for some isolated time instants of measure zero).[1] Note that $\|\phi_l(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_l + \gamma_j > \phi_{lk}(t)$, and $\mathrm{sgn}[x_{lk}(t) - x_{jk}(t)] = 1$ when $j \in \mathcal{N}_l(t)$ and $x_{jk}(t) \neq x_{lk}(t)$. Therefore, it follows from (3.7) and the fact that $\dot{x}_{lk}(t) > 0$ for all $t \in [t_1, t_2]$ almost everywhere that $\phi_{lk}(t) > 0$ and $x_{jk}(t) = x_{lk}(t)$ for all $j \in \mathcal{N}_l(t)$ and all $t \in [t_1, t_2]$. Further recall that $l \in A_{1k}(t)$ and $\dot{x}_{lk}(t) > 0$ for all $t \in [t_1, t_2]$ almost everywhere. Given these facts, there must exist two time instants $t_3 < t_4$ satisfying $[t_3, t_4] \subseteq [t_1, t_2]$ such that $\dot{x}_{jk}(t) > 0$ for all $j \in \mathcal{N}_l(t)$ and all $t \in [t_3, t_4]$ almost everywhere. Similarly, it can be obtained that $x_{qk}(t) = x_{jk}(t)$ (and hence $x_{lk}(t)$) for all $t \in [t_3, t_4]$ and all $q \in \mathcal{N}_j(t)$ with $j \in \mathcal{N}_l(t)$. Since the graph $\mathcal{G}(t)$ is connected, by induction, we have $x_{ik}(t) = x_{lk}(t)$ for all i and all t within a certain time interval, which contradicts with the assumption that $\bar{x}_k(t) \neq \underline{x}_k(t)$. Thus, $x_{ik}(t)$ for all $i \in A_{1k}(t)$ are nonincreasing when $\bar{x}_k(t) \neq \underline{x}_k(t)$. Similarly, $x_{ik}(t)$ are nondecreasing for all $i \in A_{2k}(t)$ when $\bar{x}_k(t) \neq \underline{x}_k(t)$.

To show that consensus is reached in finite time consider $V(t) = \bar{x}_k(t) - \underline{x}_k(t)$ as a Lyapunov function candidate for all $\bar{x}_k(t) \neq \underline{x}_k(t)$. Note that $V(t) > 0$ when $\bar{x}_k(t) \neq$

[1]Here, we do not consider the sets of measure zero in $[t_1, t_2]$ on which the derivatives at certain isolated time instants are nonpositive as these sets have no effect on the state value $x_{ik}(t)$.

$\underline{x}_k(t)$. Based on the above analysis, when $\bar{x}_k(t) \neq \underline{x}_k(t)$, $\dot{x}_{ik}(t) \leq 0$ for all $i \in A_{1k}(t)$ and $\dot{x}_{ik}(t) \geq 0$ for all $i \in A_{2k}(t)$ almost everywhere. Because the graph $\mathcal{G}(t)$ is connected, when $\bar{x}_k(t) \neq \underline{x}_k(t)$, there exists at least a node $\ell \in A_{1k}(t)$ having an edge to a node $j \notin A_{1k}(t)$, implying that $x_{\ell k}(t) > x_{jk}(t)$. Note that here the indices ℓ and j might change over time. It follows from (3.7) that when $\bar{x}_k(t) \neq \underline{x}_k(t)$,

$$\begin{aligned}\dot{x}_{\ell k}(t) &\leq \phi_{\ell k}(t) - [\|\phi_\ell(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_\ell + \gamma_j]\mathrm{sgn}[x_{\ell k}(t) - x_{jk}(t)] \\ &\leq -(\gamma_\ell + \gamma_j).\end{aligned}$$

Note that when $\bar{x}_k(t) \neq \underline{x}_k(t)$,

$$\begin{aligned}\dot{\bar{x}}_k(t) &= \frac{1}{|A_{1k}(t)|} \sum_{i \in A_{1k}(t)} \dot{x}_{ik}(t) \\ &= \frac{1}{|A_{1k}(t)|} [\dot{x}_{\ell k}(t) + \sum_{i \in A_{1k}(t) \setminus \{\ell\}} \dot{x}_{ik}(t)].\end{aligned}$$

Recall that when $\bar{x}_k(t) \neq \underline{x}_k(t)$, $\dot{x}_{ik}(t) \leq 0$ for all $i \in A_{1k}(t) \setminus \{\ell\}$ almost everywhere. We have $\dot{\bar{x}}_k(t) \leq -\frac{(\gamma_\ell+\gamma_j)}{|A_{1k}(t)|} \leq -\frac{2\min_{i\in\mathcal{V}}(\gamma_i)}{n-1}$ almost everywhere. Note that when $\bar{x}_k(t) \neq \underline{x}_k(t)$, $\dot{\underline{x}}_k(t) = \frac{1}{|A_{2k}(t)|}\sum_{i \in A_{2k}(t)} \dot{x}_{ik}(t) \geq 0$ almost everywhere. We hence have

$$\dot{V}(t) = \dot{\bar{x}}_k(t) - \dot{\underline{x}}_k(t) \leq -2 \min_{i \in \mathcal{V}}(\gamma_i) \Big/ (n-1),$$

almost everywhere when $\bar{x}_k(t) \neq \underline{x}_k(t)$. Based on the Lebesgue's theory for the Riemann integrability, a function on a compact interval is Riemann integrable if and only if it is bounded and the set of its discontinuous points has measure zero [5]. Therefore, although the time-derivative $\dot{V}(t)$ here is discontinuous at some time points, it is Riemann integrable. Then, we have

$$V(t) - V(0) = \int_0^t \dot{V}(\tau) d\tau \leq -\frac{2t \min_{i\in\mathcal{V}}(\gamma_i)}{n-1},$$

where $t > 0$. It follows that

$$V(t) \leq V(0) - \left[2 \min_{i \in \mathcal{V}} (\gamma_i) \Big/ (n-1) \right] t, \tag{3.8}$$

It then can be concluded that $V(t)$ converges to zero in finite time and the convergence time T satisfies $T \leq \frac{(n-1)V(0)}{2 \min_{i \in \mathcal{V}}(\gamma_i)}$. That is, consensus is reached in finite time and there exists a positive number T such that $x_i(t) = x_j(t)$ for all $t \geq T$ and all $i, j \in \mathcal{V}$. $\square$

Remark 12. *It follows from* (3.46) *that the convergence time T of the consensus process can be made smaller by selecting larger γ_i. However, if γ_i is too large, the chattering phenomenon would become worse due to the discontinuous signum function in* (3.5)*.*

Following is the main result of this section.

Theorem 13. *If Assumptions 1 to 3 hold, for the system* (3.1) *under the controller* (3.5)*, all the states $x_i(t)$ will converge asymptotically to the optimal solution $y^*(t)$ in* (3.2)*.*

Proof. Under Assumptions 1 and 2, it follows from Lemma 11 that the states of all the agents achieve consensus in finite time, i.e., there exists a time T such that $x_i(t) = x_j(t)$ for all $i, j \in \mathcal{V}$ and all $t \geq T$. For $t \geq T$, consider the Lyapunov function candidate

$$V_2(t) = \frac{1}{2} \left\{ \sum_{i=1}^{n} \nabla f_i[x_i(t), t] \right\}^T \left\{ \sum_{i=1}^{n} \nabla f_i[x_i(t), t] \right\}. \tag{3.9}$$

It follows from Theorem 3.9 in [82] that the derivative of $V_2(t)$ is,

$$\begin{aligned} \dot{V}_2(t) &= \left\{ \sum_{i=1}^{n} \nabla f_i[x_i(t), t] \right\}^T \left\{ \sum_{i=1}^{n} \frac{d \nabla f_i[x_i(t), t]}{dt} \right\} \\ &= - \left\{ \sum_{i=1}^{n} \nabla f_i[x_i(t), t] \right\}^T \left[\sum_{i=1}^{n} \Bigg(\nabla f_i[x_i(t), t] \right. \\ &\qquad \left. + \nabla^2 f_i[x_i(t), t] \sum_{j \in \mathcal{N}_i(t)} \Big\{ [\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j] \mathrm{sgn}[x_i(t) - x_j(t)] \Big\} \Bigg) \right]. \end{aligned} \tag{3.10}$$

Note from Assumptions 1 and 3 that the graph $\mathcal{G}(t)$ is undirected and $\nabla^2 f_i[x_i(t), t] = \nabla^2 f_j[x_j(t), t]$ if $x_i(t) = x_j(t)$ for all $i, j \in \mathcal{V}$, it follows that for all $t \geq T$,

$$\begin{aligned}&\sum_{i=1}^{n} \nabla^2 f_i[x_i(t), t] \sum_{j \in \mathcal{N}_i(t)} \Big\{[\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j]\text{sgn}[x_i(t) - x_j(t)]\Big\} \\ &= \nabla^2 f_i[x_i(t), t] \sum_{i=1}^{n} \sum_{j \in \mathcal{N}_i(t)} \Big\{[\|\phi_i(t)\|_\infty + \|\phi_j(t)\|_\infty + \gamma_i + \gamma_j]\text{sgn}[x_i(t) - x_j(t)]\Big\} = \mathbf{0}_m.\end{aligned}$$

Then we have for all $t \geq T$

$$\begin{aligned}\dot{V}_2(t) &= -\left\{\sum_{i=1}^{n} \nabla f_i[x_i(t), t]\right\}^T \left\{\sum_{i=1}^{n} \nabla f_i[x_i(t), t]\right\} \\ &= -2V_2(t),\end{aligned} \tag{3.11}$$

which indicates that $V_2(t) = e^{-2t}V_2(T)$ for all $t \geq T$. It can be concluded that $\lim_{t\to\infty} V_2(t) = 0$, and thus $\lim_{t\to\infty} \sum_{i=1}^{n} \nabla f_i[x_i(t), t] = \mathbf{0}_m$. Due to Assumption 3, the Lyapunov function $V_2(t)$ defined in (3.9) has a unique time-varying global minimum $y^*(t)$ such that

$$\Big\{\sum_{i=1}^{n} \nabla f_i[y^*(t), t]\Big\}^T \Big\{\sum_{i=1}^{n} \nabla f_i[y^*(t), t]\Big\} = 0.$$

Recall that $x_i(t) = x_j(t)$ for all $i, j \in \mathcal{V}$ and all $t \geq T$, which in turn implies that all $x_i(t)$ will converge to the optimal solution $y^*(t)$ in (3.2) based on Lemma 7. □

Remark 14. *In some robotic applications, it is desirable for the agents to come into a formation, while the center of the formation moves along the optimal trajectory. To achieve this goal, we introduce a deviation vector $\delta_i(t)$ for each agent i and replace $x_i(t)$ in* (3.5) *with $x_i(t) - \delta_i(t)$. It follows that Algorithm* (3.5) *will guarantee that $x_i(t) - \delta_i(t)$ converges to the optimal trajectory, which in turn implies that $x_i(t) - x_j(t)$ converges to $\delta_i(t) - \delta_j(t)$. Here, $\delta_i(t) - \delta_j(t)$ defines the desired relative position from agent j to agent i in the formation. That is, the agents will be able to converge to the optimal trajectory with the deviation vector*

$\delta_i(t)$. *The analysis follows directly by letting* $x_i(t)-\delta_i(t)$ *play the role of* $x_i(t)$ *in the previous proof.*

3.1.4 Simulations

The simulation results in this section illustrate the effectiveness of the theoretical results obtained in Sections 3.1.2. Assume that there are six agents ($n = 6$) in 2-D ($m = 2$). The network topology shown in Figure 3.1 is undirected and connected. Let $x_i(t) = [x_i^p(t), y_i^p(t)]^T \in \mathbb{R}^2$ denote the state (position) of agent i, where $x_i^p(t) \in \mathbb{R}$ (respectively, $y_i^p(t) \in \mathbb{R}$) denotes the position of agent i in the x coordinate (respectively, y coordinate).

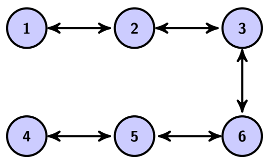

Figure 3.1: The undirected graph representing the communication topology between agents.

First, we show the simulation result using Algorithm (3.5). Consider the following unconstrained optimization problem

$$\min \sum_{i=1}^{n} \Big\{[x_i^p(t) - 0.1(0.25 + 0.5i)t]^2 + [y_i^p(t) - 0.1(0.25 + 0.5i)t]^2\Big\}. \qquad (3.12)$$

This problem is an instance of (3.2). The goal is that each state $x_i(t)$ converges to the optimal solution defined in (3.12). The intuition of the problem (3.12) is from the multi-robot target tracking problem, where $[0.1(0.25 + 0.5i)t, 0.1(0.25 + 0.5i)t]^T$ encodes the tracking target of agent i. Here, multiple robots aim to cooperatively find the optimal position that is close to all the targets. Choose $\gamma_i = 1$, $\forall i \in \mathcal{V}$. The proof of Lemma 11 proves that the maximum time for consensus to be achieved satisfies $T \leq \frac{(n-1)V(0)}{2\min_{i\in\mathcal{V}}(\gamma_i)}$. Therefore, with $\gamma_i = \gamma_j = \bar{\gamma}$, the time to achieve consensus, for any given set of initial conditions, is inversely proportional to $\bar{\gamma}$.

The initial states of the agents are chosen as $x_1(0) = [0,1]^T$, $x_2(0) = [0.5,1]^T$, $x_3(0) = [0.5,0.5]^T$, $x_4(0) = [0,0.5]^T$, $x_5(0) = [-0.5,0]^T$, $x_6(0) = [0,0]^T$. The agents' states and the optimal trajectory in the (x,t) (respectively, (y,t)) coordinates are shown in Figure 3.2(a) (respectively, (b)). The red dashed line is the optimal solution and the other solid lines are the trajectories of all agents' states. It is clear that all the agents track the optimal trajectory asymptotically (i.e, $\lim_{t\to\infty} \|x_i(t) - y^*(t)\|_2 = 0$ for all $i \in \mathcal{V}$) which is consistent with Theorem 13. We introduce a deviation vector to (3.5) by replacing x_i with $x_i - \delta_i$ (see Remark 14). Here, $\delta_1 = [0.5,0.5]^T$, $\delta_2 = [0.5,0]^T$, $\delta_3 = [0.5,-0.5]^T$, $\delta_4 = [-0.5,0.5]^T$, $\delta_5 = [-0.5,0]^T$, $\delta_6 = [-0.5,-0.5]^T$. In Figure 3.3, the blue circles present a snapshot of all the agents' initial positions and the blue crosses present two snapshots of all the agents at 4.5 s and 9 s, respectively. Figure 3.3 shows each agent's trajectories with the deviation vectors introduced (blue dashed lines), the center position of all the agents (solid black line), and the optimal trajectory (red dashed line) in the (x,y,t) coordinates. Note that the agents asymptotically form a rectangle formation with its center tracking the optimal trajectory, implying that $\lim_{t\to\infty} \|x_i(t) - \delta_i - r^*(t)\|_2 = 0$ for all $i \in \mathcal{V}$.

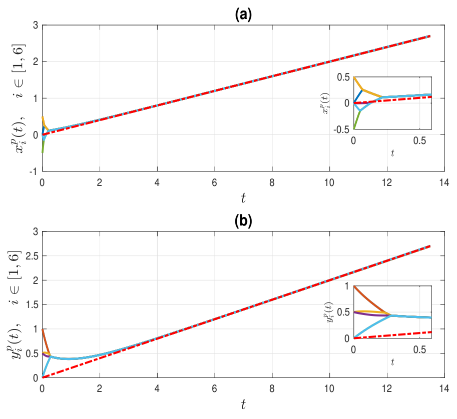

Figure 3.2: Simulation results showing state convergence using the controller (3.5).

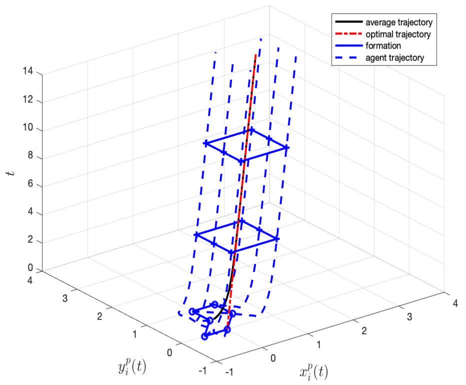

Figure 3.3: Simulation results using controller (3.5) with the deviation vectors introduced.

3.1.5 Experimental Validation

In this section, the algorithm designed in Section 3.1.2 is applied to the multi-agent formation control problem and the multi-agent moving target tracking problem and is tested in experiments. The experiments are conducted in the Cooperative Vehicle Networks (COVEN) Laboratory at the University of California, Riverside with six Crazyflie 2.0 quadrotors [78] in an 5×5 m^2 indoor environment covered by a VICON motion capture

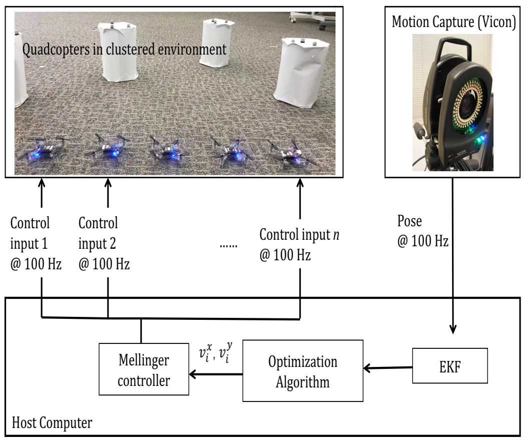

Figure 3.4: The experimental setup and information flow.

system [1]. The Crazyflies are controlled by the velocity commands (i.e., the control signals $u_i(t)$ are the velocity commands that are sent to the Crazyflies) such that their dynamics follow the single-integrator system given by (3.1). The experimental setup is illustrated in Figure 3.4. In this experiment, the control system is divided into two parts, namely, high level and low level. The high-level control involves the setup of the network topology, implementation of the distributed optimization algorithm and generation of the velocity commands $u_i(t)$. The host computer is used to run the high-level controller due to the fact that the Crazyflies used in the experiments do not have sufficient computation capability to run the controller in real time. A VICON motion capture system coupled with the Extended Kalman filter is used to estimate the positions of each agent. The host computer requests the information packet from the Vicon system every 0.01 s. The low-level control is responsible for achieving the velocity commands (using the Mellinger controller [68]). The host computer sends control commands to the Crazyflies every 0.01 s. The restrictions of a distributed environment are fully considered and the distributed network topology defined by Figure 3.1 is emulated. We establish six nodes under the robotics operating system (ROS) to control the six Crazyflies in parallel.

Multi-agent Formation Control

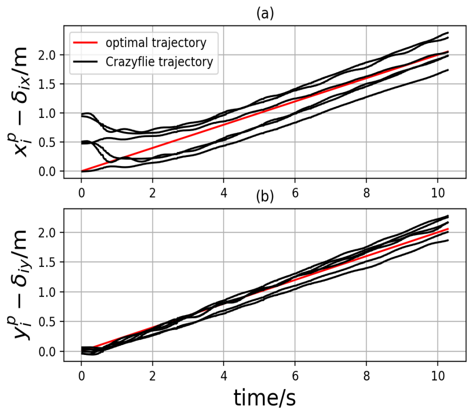

Figure 3.5: Normalized trajectory of Crazyflies using controller (3.5).

First, the distributed time-varying optimization algorithm given by (3.5) is implemented experimentally to solve the problem (3.12). The desired deviations from the optimal trajectory and the Crazyflies' initial positions have the same values as in Section V. Figure 3.5(a) (respectively, (b)) shows the six Crazyflies' normalized positions (i.e., $x_i(t) - \delta_i$) and the optimal trajectory in the (x, t) coordinate (respectively, (y, t) coordinate). The solid black lines are the normalized positions of each Crazyflie. The red solid line is the optimal trajectory. Here $\delta_i = [\delta_{ix}, \delta_{iy}]^T$. Based on Theorem 13, all $x_i(t) - \delta_i$ should converge to the optimal trajectory asymptotically, i.e., $\lim\limits_{t\to\infty} \|x_i(t) - \delta_i - r^*(t)\|_2 = 0$. This is achieved to within a tracking accuracy of 0-2cm. Various factors from the experiment might explain the tracking error: communication time-delay within the VICON system, failure to perfectly achieve the velocity commands, or interaction forces among the Crazyflies. Tracking errors

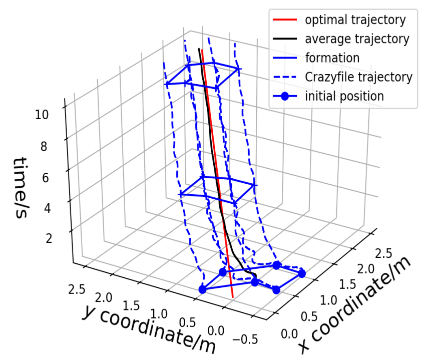

Figure 3.6: Trajectory of Crazyflies using controller (3.5).

of 2 cm are similar to those experiences in other multi-Crazyflie experiments [78]. The trajectories of all the Crazyflies (blue dashed lines), the center position of all the Crazyflies (solid black line) and the optimal trajectory (red solid line) in the (x, y, t) coordinates are shown in Figure 3.6, where the blue circles present a snapshot of all the Crazyflies' initial positions. The blue crosses present two snapshots of all the Crazyflies at 4.5 s and 9 s, respectively. As it can be seen, the center of the Crazyflies' positions tracks the optimal trajectory with small tracking errors (about 0.05cm) while the Crazyflies converge to the desired formation. When considering the center of all the Crazyflies' positions, the tracking gaps caused by the interaction forces among them should cancel.

Multi-agent Moving Target Tracking

In this subsection, we solve the moving target tracking problem using Algorithm (3.5). More precisely, the moving target tracking problem can be formulated as the following convex optimization problem:

$$\text{minimize} \qquad \frac{1}{2}\sum_{i=1}^{n} \|x_i(t) - T_i(t)\|_2^2, \tag{3.13}$$

where $x_i(t)$ is the position of robot i, and $T_i(t)$ is the position of the moving target sensed by agent i. Due to the sensing capability limitation of each agent, the position of the moving target sensed by different robots can be different. It is obvious that the optimal trajectory of problem (3.13) is $\frac{1}{n}\sum\limits_{i=1}^{n} T_i(t)$.

In our experiment, we let six Crazyflies track a moving white board (see Figure 3.4). The white board is placed on a cart that is dragged by a person to move it around. There are six marked areas located in the four corners and the middle of the two long edges on the white board. Each area is identified by three markers. The center position of each marked area (i.e., the center of three markers in the area) is sent to one assigned Crazyflie, representing the position of the moving target (white board) sensed by that Crazyflie (i.e., $T_i(t)$ in (3.13)). Essentially each Crazyflie senses a different biased position of the white board. The Crazyflies obtain their target's positions from the VICON system and calculate their targets' velocities (i.e., $\dot{T}_i(t)$ in (3.13)) based on the position data received between consecutive camera frames. We apply controller (3.5) with the same deviation vectors as those in Section V introduced to the multi-robot moving target tracking problem given by (3.13). In the experiment, we move the cart around and let the Crazyflies track the white board while maintaining the desired formation shape. Figure 3.7(a) (respectively, (b)) shows the six Crazyflies' normalized positions represented by $x_i(t) - \delta_i$ and the moving white board's center position represented by $\frac{1}{6}\sum_{i=1}^{6} T_i(t)$ in the (x, t) (respectively, (y, t)) coordinate. The black lines are the normalized positions of each Crazyflie. The red line is the center position of the white board. Figure 3.8 shows the trajectories of all the Crazyflies (blue dashed lines), the center position of all the six Crazyflies (solid black line) and the center position of the moving white board (solid red line) in the (x, y, t) coordinate, where the blue circles present a snapshot of all the Crazyflies' initial positions. The blue crosses present two snapshots of all the Crazyflies at 15 s and 28 s, respectively. It can be seen that the six Crazyflies work together to estimate and track the center position of the moving white board with small tracking errors successfully. The tracking error between each Crazyflie's actual position $x_i(t)$ and its desired position $\frac{1}{6}\sum_{i=1}^{6} T_i(t) + \delta_i$ is up to 2 cm, and the tracking error between the average trajectory of all the Crazyflies and the target's trajectory is up to 0.05 cm. The tracking errors are also acceptable.

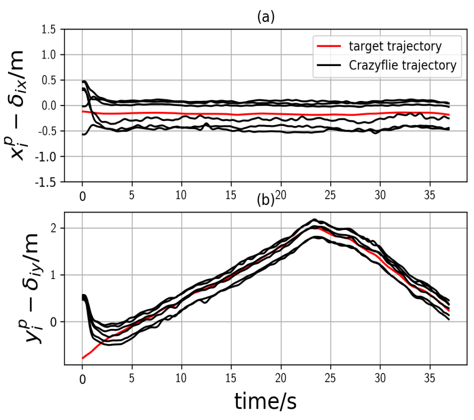

Figure 3.7: Normalized trajectory of Crazyflies in the target tracking experiment.

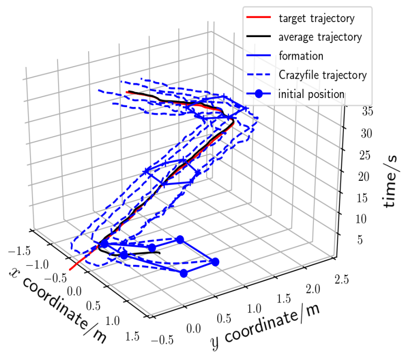

Figure 3.8: Trajectory of Crazyflies in the target tracking experiment.

3.2 Distributed Time-Varying Optimization With Equality Constraints

3.2.1 Problem Formulation

In this section, we extend the results in Section 3.1 to take into account common time-varying linear equality constraints. The goal is to design $u_i(t)$ such that all the agents cooperatively find the optimal solution $r^*(t) \in \mathbb{R}^m$ defined as

$$\begin{aligned} r^*(t) &= \underset{r(t)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} f_i[r(t), t] \right\}, \\ \text{s.t.} \qquad & A(t)r(t) = b(t), \end{aligned} \tag{3.14}$$

where $A(t) \in \mathbb{R}^{q\times m}$ and $b(t) \in \mathbb{R}^q$ are the equality constraint functions. Note that $A(t)x_i(t) = A(t)r(t)$ for all $i \in \mathcal{V}$ and $\sum_{i=1}^n f_i[x_i(t), t] = \sum_{i=1}^n f_i[r(t), t]$, if $x_i(t) = x_j(t) = r(t)$ for all $i, j \in \mathcal{V}$. Therefore, the above problem (3.14) is equivalent to finding the optimal solution $x^*(t) \in \mathbb{R}^{m*n}$ which is defined as

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} f_i[x_i(t), t] \right\}, \\ \text{s.t.} \quad A(t)x_i(t) = b(t), \quad \forall i \in \mathcal{V} \quad \text{and} \quad x_i(t) = x_j(t) \quad \forall i, j \in \mathcal{V}. \tag{3.15}$$

Here, the goal is that each state $x_i(t)$ converges to the optimal solution $r^*(t)$, i.e.,

$$\lim_{t\to\infty} [x_i(t) - r^*(t)] = \mathbf{0}_m. \tag{3.16}$$

Here agent i only has access to its own objective function $f_i[x_i(t), t]$, the constraint function $A(t)$ and $b(t)$, its own state $x_i(t)$, and information received from its neighbors $j \in \mathcal{N}_i(t)$.

We need an additional assumption.

Assumption 4. *The number of equality constraints is less than the dimension of the state variable x_i, i.e., $q < m$. Moreover, the rows of $A(t)$ are linearly independent for all $t \geq 0$, i.e.,* $\operatorname{rank}[A(t)] = q$.

Assumption 4 ensures that the constraint function has infinitely many solutions at each $t \geq 0$.

3.2.2 Algorithm Design

In this subsection, we derive a distributed control algorithm such that (3.16) holds. The Lagrangian function of problem (3.14) is

$$L[r(t), t] = \sum_{i=1}^{n} f_i[r(t), t] + \nu^T(t)[A(t)r(t) - b(t)], \tag{3.17}$$

where $\nu(t) \in \mathbb{R}^q$ is the Lagrangian multiplier. Note that the function $L[r(t), t]$ is strongly convex in $r(t)$ and concave in $\nu(t)$. Based on the KKT conditions, we know that the optimal solution of problem (3.14) must satisfy

$$\begin{aligned} \sum_{i=1}^{n} \nabla f_i[r^*(t), t] + A^T(t)\nu^*(t) = \mathbf{0}_m, \\ A(t)r^*(t) - b(t) = \mathbf{0}_q. \end{aligned} \tag{3.18}$$

Let $\lambda_i(t) \in \mathbb{R}^q$ be local internal states playing the role of the local counterparts of the global Lagrangian multiplier $\nu(t)$. Then the optimal solution in (3.18) is equivalent to

$$\lim_{t\to\infty} \|x_i(t) - x_j(t)\|_2 = 0, \ \forall i, j \in \mathcal{V}, \tag{3.19a}$$

$$\lim_{t\to\infty} \|\lambda_i(t) - \lambda_j(t)\|_2 = 0, \ \forall i, j \in \mathcal{V}, \tag{3.19b}$$

$$\sum_{i=1}^{n} \nabla f_i[x_i(t), t] + A^T(t)\lambda_i(t) = \mathbf{0}_m, \tag{3.19c}$$

$$A(t)x_i(t) - b(t) = \mathbf{0}_q, \ \forall i \in \mathcal{V}. \tag{3.19d}$$

The controller for agent i is defined as

$$\begin{aligned} u_i(t) &= \psi_i^F(t) - \sum_{j\in\mathcal{N}_i(t)} \Big\{[\|\psi_i(t)\|_\infty + \|\psi_j(t)\|_\infty + \gamma_i + \gamma_j]\mathrm{sgn}[x_i(t) - x_j(t)]\Big\}, \\ \dot{\lambda}_i(t) &= \psi_i^L(t) - \sum_{j\in\mathcal{N}_i(t)} \Big\{[\|\psi_i(t)\|_\infty + \|\psi_j(t)\|_\infty + \gamma_i + \gamma_j]\mathrm{sgn}[\lambda_i(t) - \lambda_j(t)]\Big\}, \\ \psi_i(t) &= -\Big\{\nabla^2 \tilde{L}_i[s_i(t), t]\Big\}^{-1} \Big\{\nabla \tilde{L}_i[s_i(t), t] + \frac{\partial}{\partial t}\nabla \tilde{L}_i[s_i(t), t]\Big\}, \end{aligned} \tag{3.20}$$

where $\tilde{L}_i[s_i(t), t] = f_i[x_i(t), t] + \lambda_i^T(t)[A(t)x_i(t) - b(t)]$ with $s_i(t) \in \mathbb{R}^{m+q} = [x_i^T(t), \lambda_i^T(t)]^T$, and $\psi_i^F(t)$ and $\psi_i^L(t)$ denote, respectively, the first m components and the last q components of the vector $\psi_i \in \mathbb{R}^{m+q}$. It follows from Assumptions 3 and 4 that $\nabla^2 \tilde{L}_i[s_i(t), t]$ is invertible [13]. There are four conditions in (4.17). In algorithm (3.20), the term $-\sum_{j\in\mathcal{N}_i(t)} \Big\{[\|\psi_i(t)\|_\infty + \|\psi_j(t)\|_\infty + \gamma_i + \gamma_j] \ \mathrm{sgn}[x_i(t) - x_j(t)]\Big\}$ is introduced to ensure that all the agents achieve consensus on states $x_i(t)$, i.e., the condition (3.19a). The term

$-\sum_{j\in\mathcal{N}_i(t)} \Big\{[\|\psi_i(t)\|_\infty + \|\psi_j(t)\|_\infty + \gamma_i + \gamma_j]\ \mathrm{sgn}[\lambda_i(t) - \lambda_j(t)]\Big\}$ is employed to ensure that all the agents achieve consensus on $\lambda_i(t)$, i.e., the condition (3.19b). The term ψ_i is introduced to achieve the optimal condition given by (3.19c)-(3.19d).

Remark 15. *It is worth mentioning that the discontinuous signum function in* (3.5) *and* (3.20) *might cause chattering behavior. In practice, a simple and useful way to solve this oscillating problem is to approximate the signum function using a continuous function in a region called the boundary layer around the sliding surface [26]. For example, we can replace the signum function with the function* $h(z) = \frac{z}{||z||_2+\epsilon}$, *where* $z \in \mathbb{R}^m$ *and* ϵ *is a positive constant. Despite the drawback of the chattering effect, sliding-mode control has its own merits such as fast convergence and robustness against system uncertainties and disturbances.*

3.2.3 Algorithm Analysis

This subsection establishes the asymptotic convergence of system (3.1) under controller (3.20) to the optimal solution in (3.14).

Theorem 16. *If Assumptions 1 to 4 hold, for system* (3.1) *under controller* (3.20), *then all the states* $x_i(t)$ *will converge asymptotically to the optimal solution* $r^*(t)$ *in* (3.14).

Proof. First, we show that the conditions given by (3.19a)-(3.19b) can be achieved. Applying controller (3.20) to system (3.1) leads to

$$\begin{aligned}\dot{s}_i(t) = \psi_i(t) - \sum_{j\in\mathcal{N}_i(t)} &\Big\{[\|\psi_i(t)\|_\infty + \|\psi_j(t)\|_\infty + \gamma_i + \gamma_j] \\ &\times \mathrm{sgn}[s_i(t) - s_j(t)]\Big\}.\end{aligned} \tag{3.21}$$

The desired result follows under Assumptions 1 and 2 by letting $\dot{s}_i(t)$, $s_i(t)$ and $\psi_i(t)$, respectively, play the role of $\dot{p}_i(t)$, $x_i(t)$ and $\phi_i(t)$ in the proof of Lemma 11. That is, consensus on $s_i(t)$ will be achieved in finite time. Then there exists a time T such that $s_i(t) = s_j(t)$ for all $t > T$ and all $i, j \in \mathcal{V}$ and thus $x_i(t) = x_j(t)$ and $\lambda_i(t) = \lambda_j(t)$ for all $t > T$ and all $i, j \in \mathcal{V}$.

Next we show that the conditions (3.19c)-(3.19d) will be achieved. The gradient and Hessian of the function $\tilde{L}_i[s_i(t), t]$ with respect to $s_i(t)$ are

$$\begin{aligned} \nabla \tilde{L}_i[s_i(t), t] &= \begin{bmatrix} \nabla f_i[x_i(t), t] + A^T(t)\lambda_i(t) \\ A(t)x_i(t) - b(t) \end{bmatrix}, \\ \nabla^2 \tilde{L}_i[s_i(t), t] &= \begin{bmatrix} \nabla^2 f_i[x_i(t), t] & A^T(t) \\ A(t) & \mathbf{0}_q \end{bmatrix}, \end{aligned} \tag{3.22}$$

where $\nabla^2 \tilde{L}_i[s_i(t), t]$ is invertible due to Assumptions 3 and 4. It is obvious that if $\nabla^2 f_i[x_i(t), t] = \nabla^2 f_j[x_j(t), t]$ under $x_i(t) = x_j(t)$ for all $i, j \in \mathcal{V}$, then it holds that $\nabla^2 \tilde{L}_i[s_i(t), t] = \nabla^2 \tilde{L}_j[s_j(t), t]$ under $x_i(t) = x_j(t)$ for all $i, j \in \mathcal{V}$. Consider the Lyapunov function candidate

$$V_3(t) = \frac{1}{2} \left\{ \sum_{i=1}^{n} \nabla \tilde{L}_i[s_i(t), t] \right\}^T \left\{ \sum_{i=1}^{n} \nabla \tilde{L}_i[s_i(t), t] \right\}.$$

Similar to the analysis in Theorem 13, it can be concluded that as $t \to \infty$, $V_3(t) \to 0$, we have

$$\lim_{t \to \infty} \sum_{i=1}^{n} \nabla \tilde{L}_i[s_i(t), t] = \mathbf{0}_{m+q}$$

and thus

$$\lim_{t \to \infty} \sum_{i=1}^{n} \nabla f_i[x_i(t), t] + A^T(t)\lambda_i(t) = \mathbf{0}_m$$

and

$$\lim_{t\to\infty}\sum_{i=1}^{n} A(t)x_i(t) - b(t) = \mathbf{0}_q$$

based on the definition in (3.22). The conclusion of the theorem then follows by combining the above statements. □

3.2.4 Simulations

Second, we show a simulation result using the algorithm (3.20). Let $r = [r_x, r_y]^T$ and consider the following constrained optimization problem

$$\begin{aligned} r^*(t) \in \mathbb{R}^2 = \text{argmin} \quad & \sum_{i=1}^{n} \Big\{[r_x(t) - it]^2 + [r_y(t) - it]^2\Big\}, \\ \text{s.t.} \qquad & \cos(t)r_x(t) + \sin(t)r_y(t) = 3. \end{aligned} \tag{3.23}$$

The problem (3.23) is an instance of the problem (3.14). The goal here is that each state $x_i(t)$ converges to the optimal solution $r^*(t)$ defined in (3.23). The intuition of the problem (3.23) is also from the multi-robot target tracking problem, where $[i * t, i * t]^T$ encodes the tracking signal of agent i and the function $\cos(t)r_x(t) + \sin(t)r_y(t) = 3$ represents some physical constraints for the robots. For this simulation, $\forall i \in \mathcal{V}$, we select $\gamma_i = 5$ and choose the initial states $x_i(0)$ and $y_i(0)$ randomly from the range $[-10, 10]$. The state trajectories of the agents (solid lines) and the optimal trajectory $r^*(t)$ defined in (3.23) (red dashed line) are shown in Figure 3.18. It is clear that all the agents track the optimal trajectory asymptotically, i.e, $\lim_{t\to\infty} \|x_i(t) - r^*(t)\|_2 = 0$ for all $i \in \mathcal{V}$. Figure 3.10 shows convergence of the constraint for each agent. We can see that $\cos(t)x_i(t) + \sin(t)y_i(t) - 3$ converge to zero asymptotically for all the agents, which is consistent with Theorem 16.

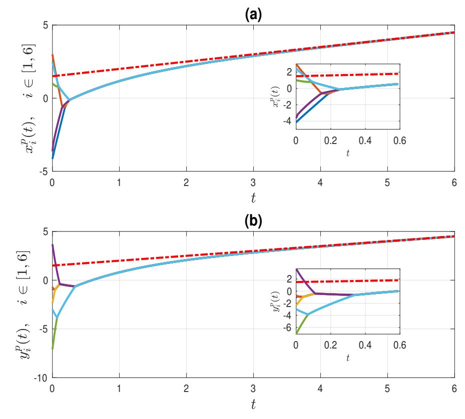

Figure 3.9: Simulation results showing state convergence to the optimal solution using controller (3.20).

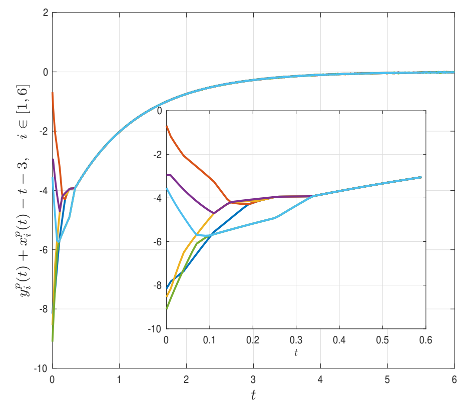

Figure 3.10: Simulation results showing convergence of the constraint using controller (3.20).

3.3 Distributed Time-Varying Optimization With Inequality Constraints

3.3.1 Problem Formulation

Consider a network consisting of n agents. Each agent is regarded as a node in an undirected graph, and each agent can only interact with its local neighbors in the network. Similary, suppose that each agent satisfies the continuous-time dynamics in (3.1). In this section, we study the distributed time-varying optimization problem with time-varying nonlinear inequality constraints. The goal is to design $u_i(t)$ using only local information and interaction, such that all the agents work together to find the optimal trajectory $\bar{y}^*(t) \in \mathbb{R}^m$ which is defined as

$$\begin{aligned} \bar{y}^*(t) = \text{argmin} \quad & \sum_{i=1}^{n} f_i[y(t), t], \\ \text{s.t.} \quad & g_i[y(t), t] \preceq \mathbf{0}_{q_i}, \ i \in \mathcal{V}, \end{aligned} \tag{3.24}$$

where $f_i[y(t), t] : \mathbb{R}^m \times \mathbb{R}_{>0} \to \mathbb{R}$ are the local objective functions, and $g_i[y(t), t] : \mathbb{R}^m \times \mathbb{R}_{>0} \to \mathbb{R}^{q_i}$ are the local inequality constraint functions. It is assumed that $f_i[y(t), t]$ and $g_i[y(t), t]$ are known only to agent i. We assume that the minimizer $\bar{y}^*(t)$ is unique for each t (see Assumption 6).

If the underlying network is connected, the above problem (3.24) is equivalent to the problem that all the agents reach consensus while optimizing the team objective function $\sum_{i=1}^{n} f_i[x_i(t), t]$ under constraints, more formally,

$$\begin{aligned} x^*(t) \in \mathbb{R}^{m*n} = \text{argmin} \quad & \sum_{i=1}^{n} f_i[x_i(t), t], \\ \text{s.t.} \quad & g_i[x_i(t), t] \preceq \mathbf{0}_{q_i}, \quad x_i(t) = x_j(t), \quad \forall i, j \in \mathcal{V}, \end{aligned} \tag{3.25}$$

where $x(t) \in \mathbb{R}^{m*n}$ is the stack of all the agents$'$ states. Here, the goal is that each state $x_i(t),\ \forall i \in \mathcal{V}$, converges to the optimal solution $\bar{y}^*(t)$, i.e.,

$$\lim_{t\to\infty} [x_i(t) - \bar{y}^*(t)] = \mathbf{0}_m. \tag{3.26}$$

Remark 17. *This architecture of the distributed time-varying constrained optimization problem* (3.24) *with networked agents finds broad applications in distributed cooperative control problems, including multi-robot navigation [54, 108] and resource allocation of power* *network [104]. For example, in a motion coordination case, knowing only their own and their neighbors′ positions, multiple UAVs might need to dock at a moving location without collision such that the total team performance is optimized. Here, the constraints can denote that the UAVs need to be located in safe areas.*

For notational simplicity, we will remove the time index t from the variables $x_i(t)$ and $u_i(t)$ in most remaining parts of this paper and only keep it in some places when necessary.

We make the following assumptions which are all standard in the literature and are used in recent works like [82, 27, 42].

Assumption 5. *The graph $\mathcal{G}$ is fixed, undirected and connected.*

Assumption 6. *All the objective functions $f_i(x_i, t)$ and the inequality constraint functions $g_i(x_i, t)$ are twice continuously differentiable with respect to x_i and continuously differentiable with respect to t. Furthermore, all the objective functions $f_i(x_i, t)$ are uniformly strongly convex in x_i, for all $t \geq 0$ and all the constraint functions $g_i(x_i, t)$ are uniformly convex in x_i, for all $t \geq 0$.*

Assumption 7. *For all $t \geq 0$, there exists at least one y such that $g_i(y, t) \prec \mathbf{0}_{q_i}$ for all $i \in \mathcal{V}$. Therefore, the Slater′s condition holds for all time.*

The uniform strong convexity of the objective functions implies that the optimal trajectory $\bar{y}^*(t)$ is unique for all $t \geq 0$ (assume it exists) if there are no constraints about the agents' states. Furthermore, we assume that $\bar{y}^*(t)$ is still unique for all $t \geq 0$ when the constraints in (3.24) are considered. By Assumption 7, the interior of the feasible region is nonempty for all $t \geq 0$ and the optimal solution $\bar{y}^*(t)$ in (3.24) at each $t \geq 0$ can be characterized using the Karush–Kuhn–Tucker (KKT) conditions.

3.3.2 Algorithm Design

In this subsection, we derive our distributed control algorithm for the time-varying constrained optimization problem in (3.25).

We design the following controller for agent i:

$$\begin{aligned} u_i &= -\beta[\nabla^2 \tilde{L}_i(x_i,t)]^{-1} \sum_{j\in\mathcal{N}_i} \operatorname{sgn}(x_i - x_j) + \phi_i(t), \\ \phi_i(t) &= -[\nabla^2 \tilde{L}_i(x_i,t)]^{-1} \left[\nabla \tilde{L}_i(x_i,t) + \frac{\partial}{\partial t}\nabla \tilde{L}_i(x_i,t)\right], \end{aligned} \tag{3.27}$$

where $\beta \in \mathbb{R}_{>0}$ is a fixed control gain, and $\tilde{L}_i(x_i,t)$ is a penalized objective function of agent i, defined as,

$$\tilde{L}_i(x_i,t) = f_i(x_i,t) - \frac{1}{\rho_i(t)} \sum_{j=1}^{q_i} \log[\sigma_i(t) - g_{ij}(x_i,t)], \tag{3.28}$$

where $g_{ij}(x_i,t) : \mathbb{R}^m \times \mathbb{R}_{>0} \to \mathbb{R}$ denotes the $j-$th component of function $g_i(x_i,t)$, $\rho_i(t) \in \mathbb{R}_{>0}$ is time-varying barrier parameter, and $\sigma_i(t) \in \mathbb{R}_{>0}$ is a time-varying slack function satisfying

$$\rho_i(t) = a_{i1}e^{a_{i2}t},\ \sigma_i(t) = a_{i3}e^{-a_{i4}t},\ a_{i1}, a_{i2}, a_{i3}, a_{i4} \in \mathbb{R}_{>0}. \tag{3.29}$$

Note that the domain of the penalized objective function $\tilde{L}_i(x_i,t)$ is $D_i = \{x_i \in \mathbb{R}^m \mid g_i(x_i,t) \prec \sigma_i(t)\mathbf{1}_{q_i}\}$. This would require that the dynamical system (3.1) with controller (3.27) is initialized at a point inside $D_i(0)$, i.e., $x_i(0) \in D_i(0)$. It is worthwhile to mention that the introduction of $\sigma_i(t)$ is to enlarge the initial feasible set. To make the algorithm (3.27) work, the initial states $x_i(0)$ need satisfy

$$g_{ij}[x_i(0), 0] < \sigma_i(0), \qquad \forall i \in \mathcal{V},\ j = 1, \cdots, q_i. \tag{3.30}$$

We will prove that the dynamical system (3.1) is well-defined under controller (3.27), initial condition (3.30), and certain other assumptions (see Lemma 24).

Remark 18. *In this work, the time-varying optimization problem* (3.25) *is deformed as a consensus subproblem and a minimization subproblem on the team objective function. We develop a distributed sliding-mode control law to address the consensus part. That is, the role of term* $-\beta[\nabla^2 \tilde{L}_i(x_i,t)]^{-1} \sum_{j \in \mathcal{N}_i} \operatorname{sgn}(x_i - x_j)$ *in* (3.27) *is to drive all the agents to reach a consensus on states* ($\lim_{t\to\infty} \|x_i(t) - \frac{1}{n}\sum_{j=1}^{n} x_j(t)\|_2 = 0$)*. Here, the Hessian-dependent gain* $\beta[\nabla^2 \tilde{L}_i(x_i,t)]^{-1}$ *is introduced to guarantee the convergence of our algorithm under nonidentical* $\nabla^2 \tilde{L}_i(x_i,t)$*. While the second term,* $\phi_i(t) \in \mathbb{R}^m$*, is an auxiliary variable playing a role in minimizing the penalized objective function* $\tilde{L}_i(x_i,t)$ *given by* (3.28)*. Note that we use the log-barrier penalty functions (see the second term in* (3.28)*) to incorporate the inequality constraints into the penalized objective function. As shown in* (3.27)*, we use the second-order/Hessian information of the penalized objective function to achieve the optimization goal.*

Remark 19. *In this work, we convert the considered constrained optimization problem into an unconstrained one using the penalty functions. Multiple penalty functions might be useful to address the inequality constraints, for example,* $\{\max[0, g_{ij}(x_i,t)]\}^2$ *and the log-barrier function used in* (3.28)*. In this work, we aim to leverage the Hessian information to solve the time-varying optimization problem. Therefore, we need a smooth and differentiable penalty function. That is why we choose log-barrier penalty functions to address the inequality constraints. While log-barrier penalty functions are not novel in its use for optimization problems with inequality constraints [111, 13, 27], our work is the first to leverage its use* *to the distributed time-varying optimization settings. Therefore, extra challenge has been introduced.*

In addition, we have

$$\nabla \tilde{L}_i(x_i,t) = \nabla f_i(x_i,t) + \sum_{j=1}^{q_i} \frac{\nabla g_{ij}(x_i,t)}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]}, \tag{3.31}$$

$$\begin{aligned}\frac{\partial}{\partial t}\nabla \tilde{L}_i(x_i,t) = {}& \frac{\partial}{\partial t}\nabla f_i(x_i,t) + \sum_{j=1}^{q_i} \frac{\partial \nabla g_{ij}(x_i,t)/\partial t}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]} - \sum_{j=1}^{q_i} \frac{\dot{\rho}_i(t)\nabla g_{ij}(x_i,t)}{\rho_i^2(t)[\sigma_i(t) - g_{ij}(x_i,t)]} \\ & - \sum_{j=1}^{q_i} \frac{\dot{\sigma}_i(t)\nabla g_{ij}(x_i,t)}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]^2} + \sum_{j=1}^{q_i} \frac{\nabla g_{ij}(x_i,t)\partial g_{ij}(x_i,t)/\partial t}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]^2},\end{aligned} \tag{3.32}$$

$$\nabla^2 \tilde{L}_i(x_i,t) = \nabla^2 f_i(x_i,t) + \sum_{j=1}^{q_i} \frac{\nabla^2 g_{ij}(x_i,t)}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]} + \sum_{j=1}^{q_i} \frac{\nabla g_{ij}(x_i,t)\nabla g_{ij}(x_i,t)^T}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]^2}, \tag{3.33}$$

where $\frac{\partial}{\partial t}\nabla f_i(x_i,t)$, $\frac{\partial}{\partial t}\nabla g_{ij}(x_i,t)$ and $\frac{\partial}{\partial t} g_{ij}(x_i,t)$ are, respectively, the partial derivatives of $\nabla f_i(x_i,t)$, $\nabla g_{ij}(x_i,t)$ and $g_{ij}(x_i,t)$ with respect to t.

Also, for notational simplicity, we will remove the time index t from the auxiliary variable $\phi_i(t)$ in most remaining parts of this paper and only keep it in some places when necessary.

Remark 20. *In this work, we convert the considered constrained optimization problem into an unconstrained one using the log-barrier penalty functions. It is worth noting that the proposed algorithm* (3.27) *is not a simple extension of the existing distributed time-varying unconstrained optimization algorithms in [82, 42]. Especially, to apply the algorithm in [42] and the consensus-based algorithm in [82] (Section III.B), it is required that the Hessians of all the local objective functions be identical. In contrast, in our context with the penalized objective functions, the Hessians of them are nonuniform due to the involvement* *of the nonuniform local constraint functions even if the original objective functions have identical Hessians. The estimator-based algorithm in [82] (Section III.C) can deal with certain objective functions with nonidentical Hessians. However, it not only necessitates the communication of certain virtual variables between neighbors with increased computation costs, but requires that the time derivatives of the Hessians and the time derivatives of the gradients of the objective functions exist and be bounded. Unfortunately, due to the complexity of the penalized objective functions in the considered constrained problem, such a requirement might be no longer guaranteed to hold and hence the result therein might not be applicable to our problem. In this paper, we introduce a novel algorithm with a Hessian-dependent gain to account for the complexity caused by the penalized objective functions, where only the partial derivatives of the gradients of the penalized objective functions with respect to time t are pre-assumed to be bounded (see Assumptions 8 and 9). Note that in [82, 42], the partial derivatives of the gradients of the objective functions with respect to time t are also required to be bounded. In this paper, we do not pre-assume that the Hessians and gradients of the penalized objective functions are bounded; however, we will prove that the Hessians and gradients of the penalized objective functions are bounded automatically under our proposed algorithms. The novel algorithm design in turn introduces new challenges in theoretical analysis, which will be addressed in the following.*

Remark 21. *In algorithm* (3.27)*, each agent just needs its own information and the relative states between itself and its neighbors. In some robotic applications, the agents' states are their spatial positions. As a result, the relative positions can be obtained by local sensing and the communication necessity might be eliminated.*

3.3.3 Algorithm Analysis

In this subsection, the asymptotical convergence of the system (3.1) under the controller (3.27) to the optimal solution in (3.24) is established. To establish our results, we require the following assumptions.

Assumption 8. *If all local states x_i are bounded, then there exists a constant $\bar{\alpha}$ such that $\sup\limits_{t\in[0,\infty)} \|\frac{\partial}{\partial t}\nabla f_i(x_i,t)\|_2 \leq \bar{\alpha}$ for all $i \in \mathcal{V}$ and $t \geq 0$.*

Assumption 9. *If all local states x_i are bounded, then there exist constants $\bar{\beta}$ and $\bar{\gamma}$ such that $\sup_{t\in[0,\infty)} \|\frac{\partial}{\partial t}\nabla g_{ij}(x_i,t)\|_2 \leq \bar{\beta}$ and $\sup_{t\in[0,\infty)} \|\frac{\partial}{\partial t} g_{ij}(x_i,t)\|_2 \leq \bar{\gamma}$, for all $i \in \mathcal{V},\ j = 1,\cdots,q_i$ and $t \geq 0$.*

Remark 22. *In Assumption 8, we assume that all $\|\frac{\partial}{\partial t}\nabla f_i(x_i,t)\|_2$ are bounded under bounded x_i. The assumption holds for an important class of situations. For example, consider the normal quadratic objective functions $\|c_i x_i + h_i(x_i,t)\|_2^2$. As long as $\frac{\partial}{\partial t}h_i(x_i,t)$ (e.g. $\sin(t), t$) are bounded under bounded x_i, $\|\frac{\partial}{\partial t}\nabla f_i(x_i,t)\|_2$ will be bounded. In Assumption 9, we assume that all $\|\frac{\partial}{\partial t}\nabla g_{ij}(x_i,t)\|_2$ and $\|\frac{\partial}{\partial t} g_{ij}(x_i,t)\|_2$ are bounded under bounded x_i. The assumption holds for an important class of situations. The boundedness of $\|\frac{\partial}{\partial t}\nabla g_{ij}(x_i,t)\|_2$ and $\|\frac{\partial}{\partial t} g_{ij}(x_i,t)\|_2$ holds for most commonly used boundary constraint functions, e.g., $x_i \leq b(t)$ or $x_i^2 \leq b(t)$ under bounded $\dot{b}(t)$.*

Remark 23. *With the piecewise-differentiable signum function involved in algorithm (3.27), the solution should be investigated in the sense of Filippov. However, since the signum function is measurable and locally essentially bounded, the Filippov solutions of the proposed system dynamics always exist [29]. To avoid symbol redundancy, we do not use the differential* *inclusions in the proofs when the Lyapunov candidates are continuously differentiable due to the following reason: if the Lyapunov function candidates are continuously differentiable, the set-valued Lie derivative of them is a singleton at the discontinuous points and the proof still holds without employing the nonsmooth analysis [22].*

In this work, we convert the considered constrained optimization problem into an unconstrained optimization problem using the log-barrier penalty functions. That the log-barrier penalty function involved in (3.28) is always well defined under our proposed algorithm is important. This is described in the next lemma.

Lemma 24. *Suppose that Assumptions 6 and 7 and the initial condition* (3.30) *hold. For the system* (3.1) *under the controller* (3.27)*, each* $x_i(t)$ *belongs to the set* $D_i = \{x_i \in \mathbb{R}^m \mid g_i(x_i, t) \prec \sigma_i(t)\mathbf{1}_{q_i}\}$ *for all* $t \geq 0$*. That is,* (3.28) *is always well defined.*

Proof. Assumption 7 ensures the existence of initial condition (3.30). Moreover, the time derivative of $\nabla \tilde{L}_i(x_i, t)$ is given by

$$\begin{aligned}\dot{\nabla} \tilde{L}_i(x_i, t) &= \frac{\partial}{\partial x_i} \nabla \tilde{L}_i(x_i, t) \times \dot{x}_i + \frac{\partial}{\partial t} \nabla \tilde{L}_i(x_i, t) \\ &= \nabla^2 \tilde{L}_i(x_i, t)\dot{x}_i + \frac{\partial}{\partial t} \nabla \tilde{L}_i(x_i, t).\end{aligned} \tag{3.34}$$

Here, Assumption 6 ensures the existence of the Hessians of the penalized functions $\tilde{L}_i(x_i, t)$, i.e., $\nabla^2 \tilde{L}_i(x_i, t)$. Substituting the solution of (3.1) with (3.27) into (3.34) leads to

$$\dot{\nabla} \tilde{L}_i(x_i, t) = -\beta \sum_{j \in \mathcal{N}_i} \text{sgn}(x_i - x_j) - \nabla \tilde{L}_i(x_i, t), \tag{3.35}$$

Then we can use the input-to-state stability [48] to analyze the system (3.35) by treating the term $-\beta \sum_{j \in \mathcal{N}_i} \text{sgn}(x_i - x_j)$ as the input and $\nabla \tilde{L}_i(x_i, t)$ as the state. Since the term

$-\beta \sum\limits_{j \in \mathcal{N}_i} \operatorname{sgn}(x_i - x_j)$ is always smaller than $n\beta$, it is obvious that each $\nabla \tilde{L}_i(x_i, t)$ remains bounded for all $t \geq 0$. Notice that (3.31) implies that $\nabla \tilde{L}_i(x_i, t)$ is unbounded at the boundary of D_i. Therefore, it follows from initial condition (3.30) that each x_i is in the set $D_i = \{x_i \in \mathbb{R}^m \mid g_i(x_i, t) \prec \sigma_i(t)\mathbf{1}_{q_i}\}$ for all $t \geq 0$. That is, (3.28) is always well defined. $\square$

In the following, in Lemma 25, we prove that the eventual states of the agents satisfy the optimal requirement shown in Lemma 7, i.e., $\lim\limits_{t\to\infty} \sum_{i=1}^n \nabla \tilde{L}_i(x_i, t) = \mathbf{0}_m$. The goal of problem (3.25) is that all the agents' states reach consensus on the optimal trajectory, and thus in Lemma 11, we prove that consensus can be achieved in finite time if all ϕ_i in the controller (3.27) are bounded, i.e., there exists a time T_2 such that $\|x_i(t) - \frac{1}{n}\sum_{j=1}^n x_j(t)\| = 0$ for all $t > T_2$ if all ϕ_i are bounded. Then in Lemma 27, we prove that all ϕ_i associated with the system (3.1) under the controller (3.27) are indeed bounded. Finally, in Theorem 13, we present that the goal in (3.26) can be achieved, i.e., $\lim\limits_{t\to\infty} \|x_i(t) - \bar{y}^*(t)\|_2 = 0$ for all $i \in \mathcal{V}$.

Lemma 25. *Suppose that Assumptions 1, 6 and 7 hold, the gain condition* (3.29) *and the initial condition* (3.30) *hold. For the system* (3.1) *under the controller* (3.27)*, the summation of all* $\nabla \tilde{L}_i(x_i, t)$ *exponentially converges to* $\mathbf{0}_m$.

Proof. It follows from Assumption 6 that all $f_i(x_i, t)$ are strongly convex in x_i. Also it follows from Assumption 6 that all $g_{ij}(x_i, t)$ are convex in x_i. From gain condition (3.29), we know that $\rho_i(t)$ and $\sigma_i(t)$ are always positive. Then it follows from initial condition (3.30) that $\tilde{L}_i(x_i, t)$ given by (3.28) must be continuously differentiable and strongly convex in x_i if x_i is in set $D_i = \{x_i \in \mathbb{R}^m \mid g_i(x_i, t) \prec \sigma_i(t)\mathbf{1}_{q_i}\}$. Note that Assumptions 6 and 7 and initial condition (3.30) hold. Lemma 24 has indicated that this is indeed the case. Therefore, each $\tilde{L}_i(x_i, t)$ must be continuously differentiable and strongly convex in x_i based on our algorithm. Consider the Lyapunov function candidate,

$$W_1 = \frac{1}{2}\left[\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)\right]^T \left[\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)\right]. \tag{3.36}$$

Note that the Lyapunov candidate W_1 is continuously differentiable. Based on the statements in Remark 23, we do not need to employ nonsmooth analysis in the stability analysis. Then we have

$$\dot{W}_1(t) = \left[\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)\right]^T \left[\sum_{i=1}^{n} \nabla^2 \tilde{L}_i(x_i, t)\dot{x}_i + \frac{\partial}{\partial t}\nabla \tilde{L}_i(x_i, t)\right]. \tag{3.37}$$

Substituting the solution of (3.1) with (3.27) into (3.37) leads to

$$\begin{aligned}\dot{W}_1(t) = \left[\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)\right]^T & \left(\sum_{i=1}^{n} \nabla^2 \tilde{L}_i(x_i, t)\left\{[\nabla^2 \tilde{L}_i(x_i, t)]^{-1}\right.\right.\\ & \left.\left.\times\beta \sum_{j\in\mathcal{N}_i} \operatorname{sgn}(x_j - x_i) + \phi_i\right\} + \frac{\partial}{\partial t}\nabla \tilde{L}_i(x_i, t)\right).\end{aligned}$$

Since the network is undirected (Assumption 5), we have $\sum\limits_{i=1}^{n} \beta \sum\limits_{j\in\mathcal{N}_i} \operatorname{sgn}(x_j - x_i) = \mathbf{0}_m$ for all $t \geq 0$. It follows that

$$\dot{W}_1(t) = \left[\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)\right]^T \left[-\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)\right] = -2W_1(t),$$

which indicates that $W_1(t) = e^{-2t}W_1(0)$ for all $t \geq 0$. It can be concluded that $W_1(t)$ exponentially converges to zero, and thus $\sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t)$ exponentially converges to $\mathbf{0}_m$.

□

Lemma 26. *Suppose that Assumptions 1, 6 and 7 hold, the gain condition* (3.29) *and the initial condition* (3.30) *hold. For the system* (3.1) *under the controller* (3.27), *if there exists* *a constant $\bar{\phi}$ such that $\sup\limits_{t\in[0,\infty)} \|\phi_i(t)\|_2 \le \bar{\phi}$, $\forall i \in \mathcal{V}$ and β satisfies that,*

$$\beta \ge \frac{2\bar{\phi} m n^2 |\mathcal{E}|}{\min_{i\in\mathcal{V}}\{\lambda_{\min}[(\nabla^2 \tilde{L}_i)^{-1}]\}} + \epsilon, ^{2} \tag{3.38}$$

where $\epsilon > 0$ is a constant, all the states x_i will achieve consensus in finite time, i.e., there exists a time T_2 such that $\|x_i(t) - x_j(t)\|_2 = 0$, for all $i, j \in \mathcal{V}$ and for all $t > T_2$.

Proof. Define

$[\nabla^2 \tilde{L}(x,t)]^{-1} = \mathrm{diag}\{[\nabla^2 \tilde{L}_1(x_1,t)]^{-1}, \cdots, [\nabla^2 \tilde{L}_n(x_n,t)]^{-1}\}$, $x = [x_1^T, \cdots, x_n^T]^T$, and $\Phi = [\phi_1^T, \cdots, \phi_n^T]^T$. Consider the Lyapunov candidate

$$W_2(t) = \|(\mathcal{D}^T \otimes I_m)x\|_1. \tag{3.39}$$

The solution of (3.1) with (3.27) can be written in compact form as

$$\dot{x} = -\beta[\nabla^2 \tilde{L}(x,t)]^{-1}(\mathcal{D} \otimes I_m)\mathrm{sgn}[(\mathcal{D}^T \otimes I_m)x] + \Phi. \tag{3.40}$$

It is obvious that $W_2(t)$ is locally Lipschitz continuous but nonsmooth at some points. Then according to Definition 4, the generalized gradient of $W_2(t)$ is given by

$$\partial W_2(t) = (\mathcal{D}^T \otimes I_m)^T \{\mathrm{SGN}[(\mathcal{D}^T \otimes I_m)x]\}, \tag{3.41}$$

where SGN$(\cdot)$ [3] is the multivalued function defined as (see Equation (20) in [22])

$$\mathrm{SGN}(z) = \begin{cases} 1 & \text{if } z > 0, \\ [-1, 1] & \text{if } z = 0, \\ -1 & \text{if } z < 0. \end{cases} \tag{3.42}$$

[2]Here $\min_{i\in\mathcal{V}}\{\lambda_{\min}[(\nabla^2 \tilde{L}_i)^{-1}]\}$ denotes the smallest value in the set $\Big[\lambda_{\min}[(\nabla^2 \tilde{L}_1)^{-1}], \lambda_{\min}[(\nabla^2 \tilde{L}_2)^{-1}], \cdots, \lambda_{\min}[(\nabla^2 \tilde{L}_n)^{-1}]\Big]$, where $\lambda_{\min}[(\nabla^2 \tilde{L}_i)^{-1}]$ is defined in Sec. II. A.

[3]With the piecewise-differentiable signum function involved in algorithm (3.27), the solution of (3.1) with (3.27) should be replaced by inclusions at a point of discontinuity.

Then based on Definition 5, the set-valued Lie derivative of $W_2(t)$ is given by

$$\dot{\tilde{W}}_2(t) = \bigcap_{\xi \in \mathrm{SGN}[(\mathcal{D}^T \otimes I_m)x]} \xi^T (\mathcal{D}^T \otimes I_m) K[f], \tag{3.43}$$

where $K[f] = \Phi - \beta[\nabla^2 \tilde{L}(x,t)]^{-1}(\mathcal{D} \otimes I_m)\mathrm{SGN}[(\mathcal{D}^T \otimes I_m)x]$ is the set-valued Filippov map of the dynamical system (3.40).

Since there is an intersection operation on the right side of (3.43), it follows that as long as $\dot{\tilde{W}}_2(t)$ is not empty and there exists $\xi \in \mathrm{SGN}[(\mathcal{D}^T \otimes I_m)x]$ such that $\xi^T(\mathcal{D}^T \otimes I_m)\tilde{f} < 0$, $\forall \tilde{f} \in K[f]$, then the result of $\dot{\tilde{W}}_2(t)$ falls into the negative half plane of the real axis. Arbitrarily choose $\eta \in \mathrm{SGN}[(\mathcal{D}^T \otimes I_m)x]$. Choose $\xi_k = \mathrm{sgn}[(\mathcal{D}^T \otimes I_m)_{k\bullet}x]$ if $\mathrm{sgn}[(\mathcal{D}^T \otimes I_m)_{k\bullet}x] \neq 0$ and $\xi_k = \eta_k$ if $\mathrm{sgn}[(\mathcal{D}^T \otimes I_m)_{k\bullet}x] = 0$, where ξ_k and η_k denote the kth element in vectors ξ and η respectively. If $\dot{\tilde{W}}_2(t) \neq \emptyset$, suppose that $\tilde{a} \in \dot{\tilde{W}}_2(t)$. It follows that

$$\begin{aligned}\tilde{a} &= -\beta\{\xi^T(\mathcal{D}^T \otimes I_m)[\nabla^2 \tilde{L}(x,t)]^{-1}(\mathcal{D} \otimes I_m)\eta\} + \xi^T(\mathcal{D}^T \otimes I_m)\Phi \\ &\leq -\beta\{\xi^T(\mathcal{D}^T \otimes I_m)[\nabla^2 \tilde{L}(x,t)]^{-1}(\mathcal{D} \otimes I_m)\xi\} + \xi^T(\mathcal{D}^T \otimes I_m)\Phi \\ &\leq -\beta\lambda_{\min}[(\nabla^2 \tilde{L})^{-1}]\|(\mathcal{D} \otimes I_m)\xi\|_2^2 + 2\bar{\phi}mn^2|\mathcal{E}|,\end{aligned} \tag{3.44}$$

If there exists an edge $(i_2, j_2) \in \mathcal{E}$ such that $x_{i_2} \neq x_{j_2}$, then $\|(\mathcal{D} \otimes I_m)\xi\| \geq 1$. It follows that

$$\tilde{a} \leq -\beta\lambda_{\min}[(\nabla^2 \tilde{L})^{-1}] + 2\bar{\phi}mn^2|\mathcal{E}|. \tag{3.45}$$

Since $\beta \geq \frac{2\bar{\phi}mn^2|\mathcal{E}|}{\min_{i\in\mathcal{V}}\{\lambda_{\min}[(\nabla^2 \tilde{L}_i)^{-1}]\}} + \epsilon = \frac{2\bar{\phi}mn^2|\mathcal{E}|}{\lambda_{\min}[(\nabla^2 \tilde{L})^{-1}]} + \epsilon$, it follows that if there exists an edge $(i_2, j_2) \in \mathcal{E}$ such that $x_{i_2} \neq x_{j_2}$, then $\tilde{a} \leq -\epsilon$. Thus, we can conclude that $\dot{\tilde{W}}_2(t) \leq -\epsilon$ if there exists an edge $(i_2, j_2) \in \mathcal{E}$ such that $x_{i_2}(t) \neq x_{j_2}(t)$. Based on the Lebesgue's theory for the Riemann integrability, a function on a compact interval is Riemann integrable if and only if it is bounded and the set of its discontinuous points has measure zero [5]. Therefore, although the time derivative $\dot{W}_2(t)$ here is discontinuous at some time points, it is Riemann integrable. Then, we have

$$W_2(t) - W_2(0) = \int_0^t \dot{W}_2(\tau) d\tau \leq -\epsilon t,$$

where $t > 0$. It follows that

$$W_2(t) \leq W_2(0) - \epsilon t, \tag{3.46}$$

Then it follows that $W_2(t)$ converges to zero in finite time and the convergence time is smaller than or equal to $W_2(0)/\epsilon$. Based on the definition of $W_2(t)$ in (3.39), we have

$$\begin{aligned} W_2(t) &= \|(\mathcal{D}^T \otimes I_m)x\|_1 \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j\in\mathcal{N}_i} \|x_i - x_j\|_1. \end{aligned} \tag{3.47}$$

That is, $W_2(t) \to 0$ implies that $\|x_i - x_j\|_1 \to 0$ for all $i \in \mathcal{V}$ and $j \in \mathcal{N}_i$. Because the network is undirected (see Assumption 5, it follows that all agents reach a consensus in finite time. That is, there exists a time T_2 such that $\|x_i(t) - \frac{1}{n}\sum_{j=1}^{n} x_j(t)\|_2 = 0$ for all $i \in \mathcal{V}$ and for all $t > T_2$. □

Lemma 27. *Suppose that Assumptions 1 and 6 to 9 hold, the gain condition* (3.29) *and the initial condition* (3.30) *hold. For the system* (3.1) *under the controller* (3.27)*, all* ϕ_i *remain bounded. That is, there exists a constant* $\bar{\phi}$ *such that* $\sup_{t\in[0,\infty)} \|\phi_i(t)\|_2 \leq \bar{\phi}$*, for all* $i \in \mathcal{V}$*.*

Proof. To begin with, we prove that each x_i associated with the system (3.1) under the controller (3.27) remains in a bounded region, which in turn guarantees that all ϕ_i are bounded. Note that Assumptions 6 and 7, the initial condition (3.30), and the gain condition (3.29) hold. Then using a similar analysis to that in Lemma 25, we have each $\tilde{L}_i(x_i, t)$ is continuously differentiable and strongly convex in x_i. The time derivative of $\nabla \tilde{L}_i(x_i,t)$ is shown in (3.35). Assume that there exists at least one x_i such that $x_i \to +\infty$ or $x_i \to -\infty$. Then due to the strongly convexity and the continuously differentiability of $\tilde{L}_i(x_i,t)$, we have $\nabla \tilde{L}_i(x_i,t) \to +\infty$ as $x_i \to +\infty$ and $\nabla \tilde{L}_i(x_i,t) \to -\infty$ as $x_i \to -\infty$. Note that Assumptions 1, 6 and 7, the initial condition (3.30), and the gain condition (3.29) hold. Then it follows from Lemma 25 that it is impossible that all x_i go to infinity at the same time. Without loss of generality, let us assume that $x_{i_1} \to +\infty$, where $i_1 = \text{argmax}_{j\in\mathcal{V}}(x_j)$. It follows that $-\beta \sum\limits_{j\in\mathcal{N}_{i_1}} \text{sgn}(x_{i_1} - x_j) \leq 0$ when $x_i \to +\infty$. Therefore, from (3.35), it is clear that $\dot{\nabla} \tilde{L}_i(x_{i_1},t)$ must be negative when $x_{i_1} \to +\infty$. Similarly, assume that $x_{i_2} \to -\infty$, where $i_2 = \text{argmin}_{j\in\mathcal{V}}(x_j)$. It follows that $-\beta \sum\limits_{j\in\mathcal{N}_{i_2}} \text{sgn}(x_{i_2} - x_j) \geq 0$ when $x_{i_2} \to -\infty$. Therefore $\dot{\nabla} \tilde{L}_i(x_{i_2},t)$ must be positive when $x_{i_2} \to -\infty$. The decreasing $\nabla \tilde{L}_i(x_i,t)$ when $x_i \to +\infty$ and increasing $\nabla \tilde{L}_i(x_i,t)$ when $x_i \to -\infty$ will result in a bounded $\nabla \tilde{L}_i(x_i,t)$ and thus a bounded x_i, which contradicts with the unbounded x_i assumption. Hence all x_i must be bounded.

Then, we will prove that all $\nabla \tilde{L}_i(x_i,t)$ are bounded for all time. It follows from Lemma 25 that $\sum\limits_{i=1}^{n} \nabla \tilde{L}_i(x_i,t)$ is always bounded. Since all x_i are bounded, we have all $\nabla f_i(x_i,t)$ and $\nabla g_{ij}(x_i,t)$ must be bounded. Then using an argument similar to Lemma 2 in [27], all $\frac{1}{\sigma_i(t) - g_{ij}(x_i,t)}$ are bounded. Therefore each $\nabla \tilde{L}_i(x_i,t)$ is always bounded for all $t \geq 0$ and for all $i \in \mathcal{V}$.

Next, we will prove that all $[\nabla^2 \tilde{L}_i(x_i,t)]^{-1}$ are bounded for all time. Since all $\tilde{L}_i(x_i,t)$ are continuous differentiable and strongly convex in its corresponding x_i, then based on the statements in Section 9.1.2 in [13], we know that all $\nabla^2 \tilde{L}_i(x_i, t)$ satisfy

$$m(t) I_n \leq \nabla^2 \tilde{L}_i(x_i, t) \leq M(t) I_n,$$

with $m(t), M(t) \in \mathbb{R}_{>0}$, which implies that all $[\nabla^2 \tilde{L}_i(x_i, t)]^{-1}$ are bounded and positive definite for all $t \geq 0$.

At last, given that all $\nabla \tilde{L}_i(x_i, t)$ and $\nabla^2 \tilde{L}_i(x_i, t)$ are bounded for all time, under Assumptions 8 and 9, it is easy to see that all $\frac{\partial}{\partial t} \nabla \tilde{L}_i(x_i, t)$ remain bounded for all $t \geq 0$.

Since $[\nabla^2 \tilde{L}_i(x_i, t)]^{-1}$, $\nabla \tilde{L}_i(x_i, t)$ and $\frac{\partial}{\partial t} \nabla \tilde{L}_i(x_i, t)$ are bounded for all $i \in \mathcal{V}$ and for all $t \geq 0$, we can get the conclusion that $\phi_i(t)$ is bounded for all $i \in \mathcal{V}$ and for all $t \geq 0$. $\square$

Theorem 28. *Suppose that Assumptions 1 and 6 to 9 hold, the initial condition* (3.30) *and the gain conditions* (3.29) *and* (3.38) *hold. For the system* (3.1) *under the controller* (3.27)*, all the states x_i will converge to the optimal solution $\bar{y}^*(t)$ in* (3.24) *eventually.*

Proof. Define

$$\tilde{y}(t)^* \in \mathbb{R}^m = \operatorname{argmin} \sum_{i=1}^{n} \tilde{L}_i[y(t), t], \tag{3.48}$$

where $\tilde{L}_i[y(t), t]$ is each agent's penalized objective function defined by (3.28). Note that Assumptions 1 and 6 to 9, initial condition (3.30) and gain condition (3.29) hold. It follows from Lemma 27 that all ϕ_i associated with the system (3.1) under the controller (3.27) are bounded for all $t \geq 0$, which in turn implies that $x_i(t) = x_j(t),\ \forall i, j \in \mathcal{V}$ in finite time according to Lemma 26. Moreover, based on Lemma 25 we know that $\lim_{t\to\infty} \sum_{i=1}^{n} \nabla \tilde{L}_i(x_i, t) = \mathbf{0}_m$. Using a similar analysis to that in Lemma 25, we have each $\tilde{L}_i(x_i, t)$ is continuously differentiable and strongly convex in x_i. Based on Lemma 7, we have

$$\sum_{i=1}^{n} \nabla \tilde{L}_i[\tilde{y}^*(t), t] = \mathbf{0}_m \tag{3.49}$$

Then it follows that all x_i will converge to the optimal solution $\tilde{y}^*(t)$ in (3.48), i.e., $\lim_{t\to\infty} x_i(t) = \tilde{y}^*(t),\ \forall i \in \mathcal{V}$.

Define

$$\begin{aligned} &\hat{y}^*(t) \in \mathbb{R}^m = \operatorname{argmin} \sum_{i=1}^{n} f_i[y(t), t] \\ &\text{s.t.} \quad g_{ij}[y(t), t] \leq \sigma_i(t),\ \forall i \in \mathcal{V}, j = 1, \cdots, q_i. \end{aligned} \tag{3.50}$$

The Lagrangian function of problem (3.50) can be written as

$$\text{Lag} = \sum_{i=1}^{n} f_i[y(t), t] + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \lambda_{ij}(t) \left\{ g_{ij}[y(t), t] - \sigma_i(t) \right\}, \tag{3.51}$$

where $\lambda_{ij}(t) > 0$ are the Lagrangian multipliers. The corresponding dual function of problem (3.50) is

$$g[\lambda_{ij}(t)] = \sup \left(\sum_{i=1}^{n} f_i[y(t), t] + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \lambda_{ij}(t) \left\{ g_{ij}[y(t), t] - \sigma_i(t) \right\} \right). \tag{3.52}$$

It follows from (3.49) and (3.31) that

$$\sum_{i=1}^{n} \nabla \tilde{L}_i[\tilde{y}^*(t), t] = \sum_{i=1}^{n} \nabla f_i[\tilde{y}^*(t), t] + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \frac{\nabla g_{ij}[\tilde{y}^*(t), t]}{\rho_i(t)\{\sigma_i(t) - g_{ij}[\tilde{y}^*(t), t]\}} = \mathbf{0}_m. \tag{3.53}$$

Define $\lambda^*_{ij}(t) = \frac{1}{\rho_i(t)\{\sigma_i(t) - g_{ij}[\tilde{y}^*(t), t]\}}$. We see that $\tilde{y}^*(t)$ minimizes the Lagrangian function of problem (3.50), which is defined in (3.51), for $\lambda_{ij}(t) = \lambda^*_{ij}(t)$. Therefore the dual function (3.52) at point $\lambda^*_{ij}(t)$ is

$$\begin{aligned} g\left[\lambda^*_{ij}(t)\right] &= \sum_{i=1}^{n} f_i[\tilde{y}^*(t), t] + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \lambda^*_{ij}(t)\{g_{ij}[\tilde{y}^*(t), t] - \sigma_i(t)\} \\ &= \sum_{i=1}^{n} f_i[\tilde{y}^*(t), t] - \sum_{i=1}^{n} \sum_{j=1}^{q_i} \frac{1}{\rho_i(t)} \\ &\leq \sum_{i=1}^{n} f_i[\hat{y}^*(t), t]. \end{aligned} \tag{3.54}$$

The last inequality in (3.54) holds since the dual function provides a lower bound to the solution of the primal problem (3.50).

It follows that

$$\left|\sum_{i=1}^{n} f_i[\hat{y}^*(t),t] - \sum_{i=1}^{n} f_i[\tilde{y}^*(t),t]\right| \leq \sum_{j=1}^{n}\sum_{k=1}^{q_j} \rho_j^{-1}(t). \tag{3.55}$$

Note that $\bar{y}^*(t) \in \mathbb{R}^m$ is the optimal solution of problem (3.24). Then we can use the perturbation and sensitivity analysis in [13] (Sec. 5.9) to analyze problem (3.50) by treating (3.50) as a perturbed version of the problem (3.24) after including the slack variables $\sigma_i(t)$ in the constraints. Under Assumption 7, the optimal solution $\bar{y}^*(t)$ can be characterized using the Karush–Kuhn–Tucker (KKT) conditions for all $t \geq 0$. Then we have

$$\left|\sum_{i=1}^{n} f_i[\hat{y}^*(t),t] - \sum_{i=1}^{n} f_i[\bar{y}^*(t),t]\right| \leq \sum_{j=1}^{n}\sum_{k=1}^{q_j} \lambda_{jk}^*(t)\sigma_j(t), \tag{3.56}$$

Hence, because $\lim_{t\to\infty} \rho_i(t) = \infty$ and $\lim_{t\to\infty} \sigma_i(t) = 0$ for all $i \in \mathcal{V}$, we have

$$\lim_{t\to\infty}\left|\sum_{i=1}^{n} f_i[\bar{y}^*(t),t] - \sum_{i=1}^{n} f_i[\tilde{y}^*(t),t]\right| = 0. \tag{3.57}$$

Since we assume that the optimal solution $\bar{y}^*(t)$ is unique, it follows that $\lim\limits_{t\to\infty} x_i(t) = \bar{y}^*(t),\ \forall i \in \mathcal{V}$. □

Remark 29. *As a byproduct, the algorithm* (3.27) *can also be used for distributed unconstrained optimization problems with much more relaxed assumptions on the objective functions (e.g., those with nonidentical Hessians) than those in [82]. In particular, it is applicable to objective functions that are strongly convex and twice continuously differentiable with respect to x_i and whose partial gradients with respect to time, i.e., $\frac{\partial \nabla f_i(x_i,t)}{\partial t}$, are bounded.*

3.3.4 Simulations

In this section, the proposed distributed time-varying constrained optimization algorithms are illustrated through two simulation cases. In both cases, we consider a network with $n = 12$ and $m = 2$. The network topology is shown by the undirected graph in Figure 3.11. Let $x_i = [x_i^p, y_i^p]^T \in \mathbb{R}^2$ denote the states of each agent. Agent i is assigned a local objective function $f_i = \frac{1}{2}[x_i^p(t) + i\sin(t)]^2 + \frac{3}{2}[y_i^p(t) - i\cos(t)]^2,\ i \in \mathcal{V}$.

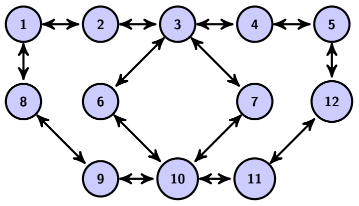

Figure 3.11: The undirected graph.

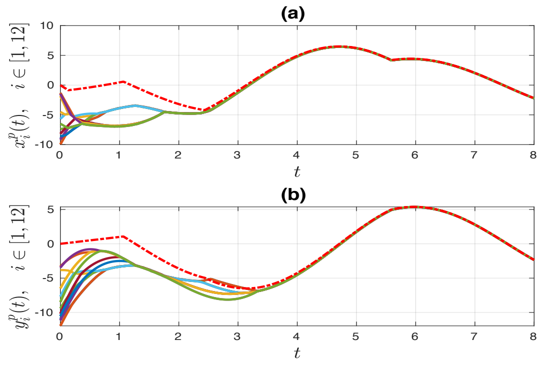

Figure 3.12: State trajectories of all the agents with the system (3.1) under the controller (3.27). The red dashed line is the optimal solution and the other solid lines are the trajectories of all agents' states.

First, we show the simulation result using the algorithm (3.27). Assume that agent j is assigned a constraint function $y_j^p(t) - x_j^p(t) - \cos(t) \leq 0$, for all $j \in [1, 2, \cdots, 6]$, and

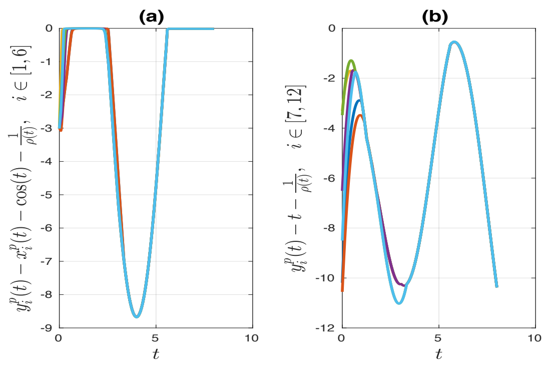

Figure 3.13: Plots of the constraint results with the system (3.1) under the controller (3.27).

agent k is assigned a constraint function $y_k^p(t) - t \leq 0$, for all $k \in [7, 8, \cdots, 12]$. All the initial states $x_i^p(0)$ are generated randomly from the range $[-10, 0]$, and $y_i^p(0) = x_i^p(0) - 2$, for all $i \in \mathcal{V}$. Therefore, the initial condition (3.30) is satisfied. We choose $\beta = 15$ and $\rho(t) = 10\exp(0.05t)$. Therefore the gain condition (3.29) is satisfied. The state trajectories of the agents are shown in Figure 3.12. We can see that all the agents track the optimal trajectory eventually which is consistent with Theorem 28. The constraint result is shown in Figure 3.13. In our simulation, agents 1–6 are assigned the constraint function $y_i^p(t) - x_i^p(t) - \cos(t) \leq 0,\ i \in [1, \cdots, 6]$, so all $y_i^p(t) - x_i^p(t) - \cos(t) - 1/\rho(t),\ i \in [1, \cdots, 6]$ always remain negative. Agents $7 - 12$ are assigned the constraint function $y_i^p(t) - t \leq 0,\ i \in [7, \cdots, 12]$, and thus all $y_i^p(t) - t - 1/\rho(t),\ i \in [7, \cdots, 12]$ always remain negative.

3.3.5 Experimental Validation

The introduced framework, distributed continuous-time time-varying constrained optimization, is of great significance in motion coordination. In this section, we apply the

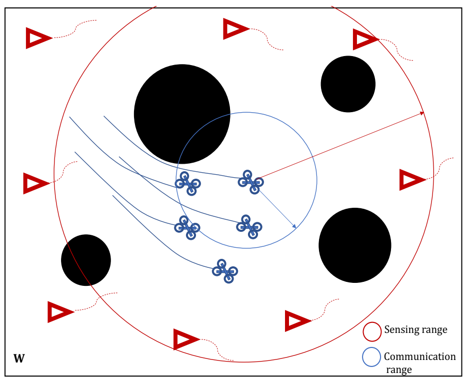

Figure 3.14: Multi-robot multi-target navigation problem

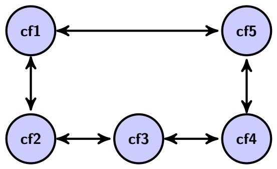

Figure 3.15: The communication topology between crazyflies.

proposed optimization algorithm (3.27) to a class of the motion coordination problems: the multi-robot multi-target navigation problem. As shown in Figure 3.14, let us consider a closed and convex workspace $W \in \mathbb{R}^2$. Consider the scenario where there are n disk-shaped robots (blue quadrotors) with center positions $x_i, i \in [1, \cdots, n]$ and radius $r_i > 0, i \in [1, \cdots, n]$ and k moving targets (red triangles) in an unknown space having obstacles inside. The objective here is to have the robots stay close while simultaneously ensuring that each independent moving target stays in the detection range of at least one robot. Assume that the workspace is populated with Q nonintersecting spherical obstacles (black circles), where the center and radius of the ith obstacle are denoted by $o_i \in W$ and $r_i^o > 0$, respectively. Since there are unknown obstacles in the environment, we have to guarantee no collisions during the tracking process.

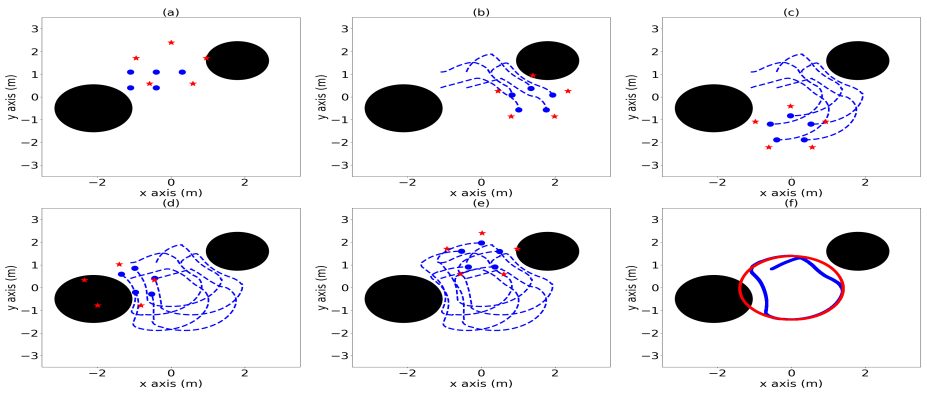

Figure 3.16: Simulation result with Crazyswarm simulator (a) Initial positions of all the crazyflies (blue circles) and all the targets (red stars). Subplots (b)-(e) show the trajectories of all the crazyflies up to time instance 25 s, 50 s, 75 s, 100 s. The positions of all the crazyflies and all the targets at each time instance are represented by blue circles (crazyflies) and red stars (targets). (f) The geometric center trajectory of all the crazyflies (blue line) and the geometric center trajecotry of all the targets (red line).

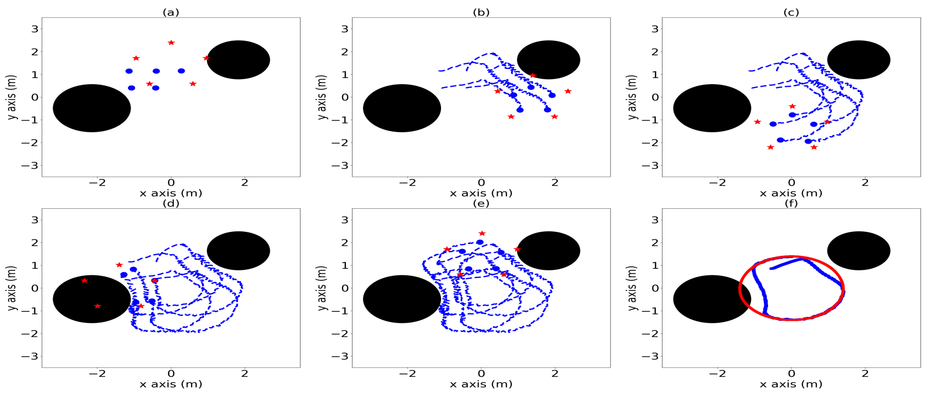

Figure 3.17: Experimental result with crazyflies (a) Initial positions of all the crazyflies (blue circles) and all the targets (red stars). Subplots (b)-(e) show the trajectories of all the crazyflies up to time instance 25 s, 50 s, 75 s, 100 s. The positions of all the crazyflies and all the targets at each time instance are represented by blue circles (crazyflies) and red stars (targets). (f) The geometric center trajectory of all the crazyflies (blue line) and the geometric center trajecotry of all the targets (red line).

We define the so-called collision-free local workspace around x_i as [27]

$$LF(x_i) = \{p \in W : a_j(x_i)^T p - b_j(x_i) \leq 0, j = 1, \cdots, Q\}, \tag{3.58}$$

where

$$\begin{aligned} a_j(x_i) &= o_j - x_i, \quad \theta_j(x_i) = \frac{1}{2} - \frac{r_j^{o2} - r_i^2}{2\|o_j - x_i\|^2}, \\ b_j(x_i) &= (o_j - x_i)^T \left[\theta_j o_j + (1-\theta_j)x_i + r_i \frac{x_i - o_j}{\|x_i - o_j\|}\right]. \end{aligned} \tag{3.59}$$

In order to have the robots stay close while simultaneously ensuring that each target stays in the sensing range of at least one robot, one method is to let all the robots assemble in the geometric center of all the targets with deviation vectors introduced to each robot. We tackle the navigation task by solving the following optimization problem with nonlinear inequality constraints,

$$\begin{aligned} \min \quad & \sum_i^n f_i = \|x_i - T_i(t)\|_2^2 \\ \text{s.t.} \quad & x_i = x_k \quad \forall i, k \in \mathcal{V}, \\ & a_j(x_i)^T x_i - b_j(x_i) \leq 0, \quad \forall i \in \mathcal{V},\ j \in [1, \cdots, q_i], \end{aligned} \tag{3.60}$$

where $T_i(t)$ is the geometric center of all the moving targets that robot i can sense and q_i is the number of obstacles that robot i can sense. Note that a robot might not be able to sense all the Q obstacles in the workspace, but it is safe enough to stay in the collision free area determined by the nearby obstacles. Since $a_j(x_i)$ and $b_j(x_i)$ depend on the position of robot i, the above optimization problem has an implicit dependence on time through x_i. However, it is very hard to directly address the inequality constraints in (3.60) due to the complexity of $a_j(x_i)$ and $b_j(x_i)$ given by (3.59). Therefore here we treat $a_j(x_i)$ and $b_j(x_i)$ as $a_j(t)$ and $b_j(t)$. Based on (3.28), the corresponding penalized objective function is defined as $\tilde{L}_i = f_i(x_i, t) - \frac{1}{\rho(t)} \sum_{j=1}^{q_i} \log\{1 - \rho(t)[a_j(t)^T x_i + b_j(t)]\}$. If the communication topology between the robots is undirected and connected, problem (3.60) satisfies all the Assumptions 1 and 6 to 9 in Theorem 28. Therefore, for robots with single-integrator dynamics defined by (3.1), the proposed constrained optimization algorithm (3.27) can be applied to reach on agreement at the geometric center of the targets and spread the robots in a desired formation about this center. Therefore, we introduce an offset vector δ_i for each robot i and replace x_i in algorithms (3.27) with $x_i - \delta_i$. Here, $\delta_i - \delta_j$ defines the desired relative position from robot j to robot i in the formation.

Our proposed algorithm is tested in the experiment with five Crazyflie 2.1 quadrotors [78] in an indoor environment. The experimental setup is the same as that in Section 3.1.5. The communication topology between the crazyflies is shown in Figure 3.15.

In our experiment, a 5×5 m^2 area is used to implement the experiment. To simplify the experiment, we assume that each crazyflie is only assigned one target moving in the environment. Note that our algorithm still works for multiple targets since we only care about the geometric center of all the targets that the crazyflie can sense. The obstacles are located at $o_1 = [-2.1 \text{ m}, -0.5 \text{ m}]$ and $o_2 = [1.8 \text{ m}, 1.6 \text{ m}]$ with radius $r_1^0 = 0.9$ m, and $r_2^0 = 0.7$ m. Each crazyflie is able to sense an obstacle if any point of the obstacle falls into the circle with the center being the crazyflie position and the radius being 1.0 m. The offset vectors are chosen as

$$\delta_1 = [0.2\sin(0.2\pi)\ \mathrm{m}, -0.2\cos(0.2\pi)\ \mathrm{m}]^T,$$

$$\delta_2 = [-0.2\sin(0.2\pi)\ \mathrm{m}, -0.2\cos(0.2\pi)\ \mathrm{m}]^T,$$

$$\delta_3 = [-0.2\cos(0.1\pi)\ \mathrm{m}, 0.2\sin(0.1\pi)\ \mathrm{m}]^T,$$

$$\delta_4 = [0\ \mathrm{m}, 0.2\ \mathrm{m}]^T,$$

$$\delta_5 = [0.2\cos(0.1\pi)\ \mathrm{m}, 0.2\sin(0.1\pi)\ \mathrm{m}]^T.$$

The initial positions of the five crazyflies are chosen as $x_1(0) = [-0.4\ \mathrm{m}, 0.4\ \mathrm{m}]$, $x_2(0) = [-1.1\,\mathrm{m}, 0.4\,\mathrm{m}]$, $x_3(0) = [-1.1\,\mathrm{m}, 1.1\,\mathrm{m}]$, $x_4(0) = [-0.4\,\mathrm{m}, 1.1\,\mathrm{m}]$, and $x_5(0) = [0.3\,\mathrm{m}, 1.1\,\mathrm{m}]$. We choose $\rho(t) = 125\exp(0.01t)$ and $\beta = 5$. The trajectories of the crazyflies in the Crazyswarm simulator [78] and in the experiment are, respectively, shown in Figures 3.16 and 3.17. In both figures, the black circles are obstacles and the blue lines are the trajectories of the crazyflies. Subplots (a)-(e) show the trajectories of all the crazyflies up to time instances 0 s, 25 s, 50 s, 75 s and 100 s. In addition, five snapshots at 0 s, 25 s, 50 s, 75 s and 100 s denoted by the red stars (targets) and blue circels (crazyflies) are shown in subplots (a)-(e). It is obvious that all the crazyflies assemble together and avoid obstacles successfully both in the Crazyswarm simulator and real experiment. Subplot (f) shows the trajectories of the geometric center of all the crazyflies (blue line) and all the targets (red line). In the crazywarm simulator, the geometric center of all crazyflies are able to track the geometric center of all the targets with zero tracking error which are consistent with Theorem 28. In our experimental result, the crazyflies tremble slightly in flight and the geometric center of all crazyflies are able to track the geometric center of all the targets with small tracking error (about 0.001 m). It is worthwhile to mention that the trembling phenomena and tracking error in the experiment might stem from the time-delay of communication with the Vicon system and failure of achieving the velocity commands accurately.

3.4 Distributed Time-Varying Optimization With Both Inequality and Equality Constraints

3.4.1 Problem Formulation

In this section, we extend the results in Section III.A to take into account both time-varying nonlinear inequality and linear equality constraints. The goal is to design $u_i(t)$ using only local information and local interaction for the system (3.1), such that all the agents work together to find the optimal trajectory $r^*(t) \in \mathbb{R}^m$ defined as

$$\begin{aligned} \bar{r}^*(t) &= \text{argmin} \quad \sum_{i=1}^{n} f_i[r(t), t], \\ &\text{s.t.} \quad g_i[r(t), t] \preceq \mathbf{0}_{q_i}, \quad A_i(t) r(t) = b_i(t), \quad \forall i \in \mathcal{V}, \end{aligned} \tag{3.61}$$

where $A_i(t) \in \mathbb{R}^{p_i \times m}$ and $b_i(t) \in \mathbb{R}^{p_i}$ are the local equality constraint functions. It is assumed that $A_i(t)$ and $b_i(t)$ are known only to agent i and are continuously differentiable with respect to t. Here the goal is that each state $x_i(t)$ converges to the optimal solution $\bar{r}^*(t)$, i.e.,

$$\lim_{t \to \infty} [x_i(t) - \bar{r}^*(t)] = \mathbf{0}_m \tag{3.62}$$

We need an additional assumption.

Assumption 10. *The number of the equality constraints is less than the dimension of the agents′ states, i.e. $p_i < m$, and rank$(A_i) = p_i$, for all $i \in \mathcal{V}$. And for all $t \geq 0$, there exists at least one r such that $A_i(t) r = b_i(t)$ for all $i \in \mathcal{V}$.*

Assumption 10 ensures that the system of equations $A_i(t)x_i(t) = b_i(t)$ is consistent and has infinitely many solutions at each $t \geq 0$. We assume that the optimal solution $\bar{r}^*(t)$ in (3.61) is unique for all $t \geq 0$. For notational simplicity, we will remove the time index t from the variables $A_i(t)$ and $b_i(t)$ in most remaining parts of this paper and only keep it in some places when necessary.

3.4.2 Algorithm Design

In this subsection, we derive a distributed control algorithm such that (3.62) holds. In addition, in the algorithm (3.27), it is required that the upper bounds on auxiliary variable ϕ_i be known in advance such that the control gain β can be chosen to satisfy (3.38). To remove this restriction, we introduce an adaptive gain design in the algorithm.

We design the following controller for agent i:

$$u_i = -w_i(t)[\nabla^2 \hat{L}_i(x_i,t)]^{-1} \sum_{j\in\mathcal{N}_i} \operatorname{sgn}(x_i - x_j) + \phi_i, \tag{3.63a}$$

$$\phi_i = -[\nabla^2 \hat{L}_i(x_i,t)]^{-1} \left[\nabla \hat{L}_i(x_i,t) + \frac{\partial}{\partial t}\nabla \hat{L}_i(x_i,t)\right], \tag{3.63b}$$

$$\dot{s}_i(t) = \sum_{j\in\mathcal{N}_i} \operatorname{sgn}(\|x_i - x_j\|_1), \tag{3.63c}$$

$$w_i(t) = z_i(t) + s_i(t), \tag{3.63d}$$

$$\dot{z}_i(t) = -\alpha \sum_{j\in\mathcal{N}_i} \operatorname{sgn}[w_i(t) - w_j(t)]. \tag{3.63e}$$

In (3.63a) and (3.63b), $w_i(t) \in \mathbb{R}$ is a dynamic gain, and $\hat{L}_i(x_i,t)$ is the penalized objective function of agent i given by

$$\hat{L}_i(x_i,t) = f_i(x_i,t) - \frac{1}{\rho_i(t)} \sum_{j=1}^{q_i} \log[\sigma_i(t) - g_{ij}(x_i,t)] + \frac{\kappa_i}{2}\|A_i x_i - b_i\|_2^2, \tag{3.64}$$

where $\kappa_i \in \mathbb{R}_{>0}$ is a constant gain. Note that (3.64) includes the local log-barrier penalty functions and the local quadratic penalty functions to account for, respectively, the inequality and equality constraints in (3.61). In (3.63c), the gain $s_i(t) \in \mathbb{R}$ is adapted according to the state differences between agent i and its neighbors. The dynamic gain $w_i(t)$ is the output of a distributed average tracking estimator given by (3.63d)) and (3.63e), where $\alpha \in \mathbb{R}_{>0}$ is a constant gain, $z_i(t) \in \mathbb{R}$ is the internal state, and $s_i(t)$ is the reference signal associated with agent i. In the next subsection, we will show that the dynamic gain $w_i(t)$ can help all the agents achieve consensus without knowing certain prior information. In addition, we have

$$\nabla \hat{L}_i(x_i,t) = \nabla f_i(x_i,t) + \sum_{j=1}^{q_i} \frac{\nabla g_{ij}(x_i,t)}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]} + \kappa_i A_i^T(A_i x_i - b_i), \tag{3.65}$$

$$\begin{aligned} \frac{\partial}{\partial t}\nabla \hat{L}_i(x_i,t) = {} & \frac{\partial}{\partial t}\nabla f_i(x_i,t) + \sum_{j=1}^{q_i} \frac{\partial \nabla g_{ij}(x_i,t)/\partial t}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]} - \sum_{j=1}^{q_i} \frac{\dot{\rho}_i(t)\nabla g_{ij}(x_i,t)}{\rho_i^2(t)[\sigma_i(t) - g_{ij}(x_i,t)]} \\ & - \sum_{j=1}^{q_i} \frac{\dot{\sigma}_i(t)\nabla g_{ij}(x_i,t)}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]^2} + \sum_{j=1}^{q_i} \frac{\nabla g_{ij}(x_i,t)\partial g_{ij}(x_i,t)/\partial t}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]^2} \\ & + 2\kappa_i A_i^T \dot{A}_i x_i - \kappa_i \dot{A}_i b_i - \kappa_i A_i \dot{b}_i, \end{aligned} \tag{3.66}$$

$$\begin{aligned} \nabla^2 \hat{L}_i(x_i,t) = {} & \nabla^2 f_i(x_i,t) + \sum_{j=1}^{q_i} \frac{\nabla^2 g_{ij}(x_i,t)}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]} \\ & + \sum_{j=1}^{q_i} \frac{\nabla g_{ij}(x_i,t)\nabla g_{ij}(x_i,t)^T}{\rho_i(t)[\sigma_i(t) - g_{ij}(x_i,t)]^2} + \kappa_i A_i^T A_i, \end{aligned} \tag{3.67}$$

To make the algorithm (3.63) work, the gain α, and the initial internal states $s_i(0)$ and $z_i(0)$ need satisfy

$$\alpha > n, \tag{3.68}$$

$$s_i(0) > 0, \quad \forall i \in \mathcal{V}, \tag{3.69}$$

$$z_i(0) = 0, \quad \forall i \in \mathcal{V}. \tag{3.70}$$

Remark 30. *In the algorithm* (3.27)*, it is required that the bounds on the auxiliary variables ϕ_i be known in advance such that the control gain β can be chosen to satisfy* (3.38)*. However, these bounds might not be obtained or estimated accurately in certain circumstances. Therefore, in the algorithm* (3.63)*, adaptive control gains are designed to remove the need for using the information of these bounds. The tradeoff is that the virtual variable $w_i(t)$ need to be communicated between neighbors to implement the algorithm* (3.63)*.*

Remark 31. *Algorithm* (3.63) *is distributed since each agent only needs its own information and information received from its neighbors. Take agent i as an example. Agent i uses it own information: $x_i(t)$, $w_i(t)$, $s_i(t)$, $z_i(t)$, and Hessian and gradient information of its penalized objective function. It is worthwhile to mention that agent i only needs to know its own penalty parameters: $\rho_i(t)$, $\sigma_i(t)$ and $\kappa_i(t)$. Moreover, agent i needs to know information received from it neighbors: $x_j(t)$ and $w_j(t)$, $j \in \mathcal{N}_i$. While a common control gain α is needed for all the agents, α is a constant and only required to be larger than n. Ref. [18] provides an answer about how to estimate n in a distributed way.*

The estimator given by (3.63d) and (3.63e) guarantees that the gains $w_i(t)$ for all agents become uniform after a finite time as shown in the following lemma.

Lemma 32. *Suppose that Assumption 5 hold, the gain condition* (4.23) *and the initial condition* (3.70) *hold. For the system* (3.1) *under the controller* (3.63)*, all $w_i(t)$ will converge to $\frac{1}{n}\sum_{j=1}^{n} s_j(t)$ in finite time. That is there exists a time T_0 such that $w_i(t) = \frac{1}{n}\sum_{i=j}^{n} s_j(t)$ for all $i \in \mathcal{V}$, and $t \geq T_0$.*

Proof. The proof is evident based on Theorem 1 in [15]. $\square$

In the following, for notational simplicity, we will remove the time index t from $s_i(t)$, $z_i(t)$, and $w_i(t)$ in most remaining parts of this paper and only keep it in some places when necessary.

3.4.3 Algorithm Analysis

In this subsection, the asymptotical convergence of the system (3.1) under the controller (3.63) to the vicinity of the optimal solution in (3.61) is established. Note that in the case where there only exist nonlinear inequality constraints, the algorithm (3.27) is capable of tracking the optimal solution in (3.24) with a zero tracking error. Since we use the quadratic penalty functions to account for the equality constraints, the larger the penalty weight κ, the better the approximation x_i to a solution of the original problem (3.61). We need an additional assumption.

Assumption 11. *The time derivatives of the local constraint parameters are bounded. That is, there exist constants $\bar{a}$ and $\bar{b}$ such that $\sup_{t\in[0,\infty)} \|\dot{A}_i(t)\|_2 \leq \bar{a}$ and $\sup_{t\in[0,\infty)} \|\dot{b}_i(t)\|_2 \leq \bar{b}$, for all $i \in \mathcal{V}$, and for all $t \geq 0$.*

In the following, we give the main results on the distributed continuous-time optimization with time-varying nonlinear inequality and linear equality constraints.

Theorem 33. *Suppose that Assumptions 1, 4, 6 to 9 and 11 hold, the initial conditions* (3.30), (3.69), *and* (3.70) *hold, and the gain conditions* (3.29) *and* (4.23) *hold. For the system* (3.1) *under the controller* (3.63), *all the states x_i will converge to the vicinity of the optimal solution $\bar{r}^*(t)$ in* (3.61) *eventually.*

Proof. We first show that (3.64) is always well defined under our proposed algorithm (3.63).

The time derivative of $\nabla \hat{L}_i(x_i,t)$ is given by

$$\dot{\nabla}\hat{L}_i(x_i,t) = \nabla^2 \hat{L}_i(x_i,t)\dot{x}_i + \frac{\partial}{\partial t}\nabla \hat{L}_i(x_i,t). \tag{3.71}$$

Substituting the solution of (3.1) with (3.63) into (3.71) leads to

$$\dot{\nabla}\hat{L}_i(x_i,t) = -w_i \sum_{j\in\mathcal{N}_i} \operatorname{sgn}(x_i - x_j) - \nabla \hat{L}_i(x_i,t), \tag{3.72}$$

Then using a similar analysis to that in Lemma 24, we have that each $\nabla \hat{L}_i(x_i,t)$ must remain bounded for all time and thus each x_i is in the set $D_i = \{x_i \in \mathbb{R}^m \mid g_i(x_i,t) \prec \sigma_i(t)\mathbf{1}_{q_i}\}$ for all $t \geq 0$. That is, (3.65) is well defined for all $t \geq 0$.

Then we show that the agents' states under the controller (3.63) satisfy the optimal requirement shown in Lemma 7 eventually. It follows from Assumption 6 that all $f_i(x_i,t)$ are strongly convex in x_i. Also it follows from Assumption 6 that all $g_{ij}(x_i,t)$ are convex in x_i. From gain condition (3.29), we know that $\rho_i(t)$ and $\sigma_i(t)$ are always positive. In addition, from Assumption 10, we know that $A_i^T A_i$ must be positive semidefinite. Note that κ_i is a positive constant. Therefore, from initial condition (3.30) we know that $\hat{L}_i(x_i,t)$ given by (3.64) must be continuously differentiable and strongly convex in x_i if x_i is in the set $D_i = \{x_i \in \mathbb{R}^m \mid g_i(x_i,t) \prec \sigma_i(t)\mathbf{1}_{q_i}\}$. The above analysis has indicated that this is indeed the case. Therefore, each $\hat{L}_i(x_i,t)$ must be continuously differentiable and strongly convex in x_i based on our algorithm (3.63). Consider the Lyapunov function candidate,

$$W_3 = \frac{1}{2}\left[\sum_{i=1}^{n} \nabla \hat{L}_i(x_i,t)\right]^T \left[\sum_{i=1}^{n} \nabla \hat{L}_i(x_i,t)\right]. \tag{3.73}$$

Similarly, based on the statements in Remark 23, there is no need to employ the nonsmooth analysis. Then we have

$$\dot{W}_3(t) = \left[\sum_{i=1}^{n} \nabla \hat{L}_i(x_i,t)\right]^T \left[\sum_{i=1}^{n} \nabla^2 \hat{L}_i(x_i,t)\dot{x}_i + \frac{\partial}{\partial t}\nabla \hat{L}_i(x_i,t)\right]. \tag{3.74}$$

Substituting the solution of (3.1) with (3.63) into (3.74) leads to

$$\dot{W}_3(t) = \left[\sum_{i=1}^{n} \nabla \hat{L}_i(x_i, t)\right]^T \left(\sum_{i=1}^{n} \nabla^2 \hat{L}_i(x_i, t) \left\{[\nabla^2 \hat{L}_i(x_i, t)]^{-1} \right.\right.$$
$$\left.\left. \times w_i \sum_{j \in \mathcal{N}_i} \operatorname{sgn}(x_j - x_i) + \phi_i \right\} + \frac{\partial}{\partial t} \nabla \hat{L}_i(x_i, t)\right).$$

Notice that Assumption 5, the initial condition (3.70), and the gain condition (4.23) hold, it follows from Lemma 32 that $w_i = w_j$, for all $i, j \in \mathcal{V}$ and for all $t \geq T_0$. Since the network is undirected (Assumption 5), we have $\sum\limits_{i=1}^{n} w_i \sum\limits_{j \in \mathcal{N}_i} \operatorname{sgn}(x_j - x_i) = \mathbf{0}_m$ for all $t \geq T_0$. Then for all $t \geq T_0$, we have

$$\dot{W}_3(t) = \left[\sum_{i=1}^{n} \nabla \hat{L}_i(x_i, t)\right]^T \left[-\sum_{i=1}^{n} \nabla \hat{L}_i(x_i, t)\right] = -2W_3(t),$$

indicating that $W_3(t) = e^{-2(t-T_0)} W_3(T_0)$ for all $t \geq T_0$. The time derivative of $\nabla \hat{L}_i(x_i, t)$ is given by (3.71). It follows that each $\nabla \hat{L}_i(x_i, t)$ is bounded at all time. Therefore, $W_3(T_0)$ is bounded. Then it can be concluded that $W_3(t)$ exponentially converges to zero, and thus $\sum_{i=1}^{n} \nabla \hat{L}_i(x_i, t)$ exponentially converges to $\mathbf{0}_m$.

Next, we show that all x_i remain bounded under the algorithm (3.63). Based on (3.63c)-(3.63e), the time derivative of w_i is given by

$$\begin{aligned} \dot{w}_i &= \dot{z}_i + \dot{s}_i \\ &= -\alpha \sum_{j \in \mathcal{N}_i} \operatorname{sgn}(w_i - w_j) + \sum_{j \in \mathcal{N}_i} \operatorname{sgn}(\|x_i - x_j\|_1). \end{aligned}$$

It follows that $\dot{w}_{\min}$ must be non-negative, where $w_{\min}$ is defined as $\min\limits_i w_i$. The reason is that $\operatorname{sgn}(w_{\min} - w_j)$ must be non-positive. Note that $w_i(0) = z_i(0) + s_i(0)$. It follows from (3.69) and (3.70) that $w_i(0) > 0$, $\forall i \in \mathcal{V}$, which in turn guarantees that $w_{\min}$ and thus all w_i are positive for all $t \geq 0$. Then similar to the proof in Lemma 27, we have that all x_i

remain bounded for all $t \geq 0$. Given the above results and Assumption 11, using similar analysis to that in Lemma 27, it is easy to prove that each $\phi_i(t)$ is bounded for all the time.

Next, we show that all the agents reach a consensus in finite time. Consider any edge $(i, j) \in \mathcal{E}$. Let $0 < t_{11}^{ij} < t_{12}^{ij} < t_{21}^{ij} < t_{22}^{ij} < \cdots$ denote the contiguous switching times such that $x_i \neq x_j$ during the time interval $[t_{k1}^{ij}, t_{k2}^{ij}]$ and $x_i = x_j$ during the time interval $[t_{k2}^{ij}, t_{k+1,1}^{ij})$, $k = 1, 2, \cdots$. From the dynamics of s_i in (3.63c), it is easy to see that $s_i(\infty) = \sum_{j \in \mathcal{N}_i} \sum_{k=1}^{\infty} (t_{k2}^{ij} - t_{k1}^{ij}) + s_i(0)$. It follows from Lemma 32 that $w_i(t) = w_j(t) = \frac{1}{n} \sum_{k=1}^{n} s_k(t)$ for all $i, j \in \mathcal{V}$ and for all $t \geq T_0$. If for all edges, $\sum_{k=1}^{\infty} (t_{k2}^{ij} - t_{k1}^{ij}) < \infty$, $\forall (i, j) \in \mathcal{E}$, it is clear that $t_{k2}^{ij} - t_{k1}^{ij} \to 0$ as $k \to \infty$. Since the graph is connected (Assumption 1), it follows that consensus can be achieved eventually. If there exists an edge (i, j) such that $\sum_{k=1}^{\infty} (t_{k2}^{ij} - t_{k1}^{ij}) = \infty$, then we have $s_i(\infty) = \infty$ and $w_i(\infty) = w_j(\infty) = \infty$ for all $i, j \in \mathcal{V}$. Then there must exist a time $T_1 > T_0$ such that $w_i(T_1) = w_j(T_1) > \frac{2\bar{\phi} m n^2 |\mathcal{E}|}{\min_{i \in \mathcal{V}} \{\lambda_{\min}[(\nabla^2 \hat{L}_i)^{-1}]\}}$ for all $i, j \in \mathcal{V}$ and all $t \geq T_1$. Then similar to the proof of Lemma 26, we have that all agents reach a consensus in finite time, i.e., there exists a time T_2 such that $\|x_i(t) - x_j(t)\|_2 = 0$ for all $t > T_2$.

Now, we show that all the agents with the system (3.1) under the controller (3.63) converge to the vicinity of the optimal solution $\bar{r}^*(t)$ in (3.61). Define

$$\tilde{r}^*(t) \in \mathbb{R}^m = \operatorname{argmin} \sum_{i=1}^{n} \hat{L}_i[r(t), t],$$

where $\hat{L}_i[r(t), t]$ is each agent's penalized objective function defined by (3.64). Summarizing the above analysis and similar to the analysis in Theorem 28, it follows from Lemma 7 that all x_i converge to the optimal solution $\tilde{r}^*(t)$, i.e., $\lim_{t \to \infty} x_i(t) = \tilde{r}^*(t)$, $\forall i \in \mathcal{V}$.

Define

$$\hat{r}^*(t) \in \mathbb{R}^m = \operatorname{argmin} \sum_{i=1}^{n} f_i[r(t), t] + \frac{\kappa_i}{2} \|A_i r(t) - b_i\|_2^2$$

$$\text{s.t.} \quad g_{ij}[r(t), t] \leq \sigma_i(t), \ \forall i \in \mathcal{V}, j = 1, \cdots, q_i.$$

Similar to the analysis in Theorem 28, we know that

$$\left| \sum_{i=1}^{n} \left\{ f_i[\hat{r}^*(t), t] + \frac{\kappa_i}{2} \|A_i \hat{r}^*(t) - b_i\|_2^2 \right\} - \sum_{i=1}^{n} \left\{ f_i[\tilde{r}^*(t), t] + \frac{\kappa_i}{2} \|A_i \tilde{r}^*(t) - b_i\|_2^2 \right\} \right|$$
$$\leq \sum_{j=1}^{n} \sum_{k=1}^{q_j} \rho_j^{-1}(t).$$

Define

$$\bar{\bar{r}}(t)^* \in \mathbb{R}^m = \operatorname{argmin} \sum_{i=1}^{n} f_i[r(t), t] + \frac{\kappa_i}{2} \|A_i r(t) - b_i\|_2^2$$
$$\text{s.t.} \qquad g_{ij}(x_i, t) \leq 0, \ \forall i \in \mathcal{V}, j = 1, \cdots, q_i. \tag{3.75}$$

Then based on [13] (Sec. 5.9), we have

$$\left| \sum_{i=1}^{n} \left\{ f_i[\hat{r}^*(t), t] + \frac{\kappa_i}{2} \|A_i \hat{r}^*(t) - b_i\|_2^2 \right\} - \sum_{i=1}^{n} \left\{ f_i[\bar{\bar{r}}^*(t), t] + \frac{\kappa_i}{2} \|A_i \bar{\bar{r}}^*(t) - b_i\|_2^2 \right\} \right|$$
$$\leq \sum_{j=1}^{n} \sum_{k=1}^{q_j} \lambda_{jk}^*(t) \sigma_j(t),$$

where $\lambda_{jk}(t)$ are the Lagrangian multipliers corresponding to the inequality constraint defined in (3.75), and $\lambda_{jk}^*(t)$ are the optimal Lagrangian multipliers. Hence, because $\lim_{t\to\infty} \rho_i(t) = \infty$ and $\lim_{t\to\infty} \sigma_i(t) = 0$ for all $i \in \mathcal{V}$, we have

$$\lim_{t\to\infty} \left| \sum_{i=1}^{n} \left\{ f_i[\bar{\bar{r}}^*(t), t] + \frac{\kappa_i}{2} \|A_i \bar{\bar{r}}^*(t) - b_i\|_2^2 \right\} - \sum_{i=1}^{n} \left\{ f_i[\tilde{r}^*(t), t] + \frac{\kappa_i}{2} \|A_i \tilde{r}^*(t) - b_i\|_2^2 \right\} \right| = 0,$$

which indicates that $\lim\limits_{t\to\infty} x_i(t) = \bar{\bar{r}}^*(t), \ \forall i \in \mathcal{V}$. Then based on the standard quadratic penalty theory [13], $\bar{\bar{r}}^*(t)$ is in the neighborhood of the optimal solution $\bar{r}^*(t) \in \mathbb{R}^m$ in (3.61). The conclusion of the theorem then follows by combining the above statements. $\square$

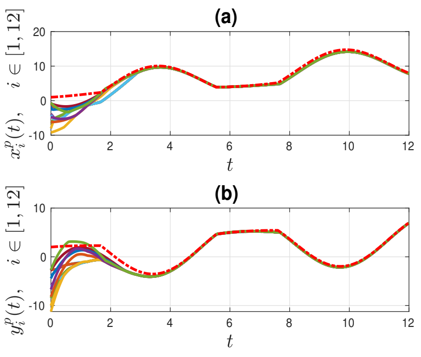

Figure 3.18: State trajectories of all the agents with the system (3.1) under the controller (3.63). The red dashed line is the optimal solution and the other solid lines are the trajectories of all agents' states.

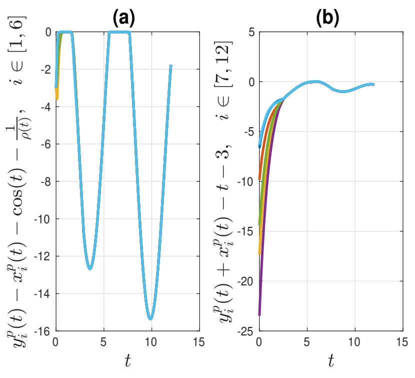

Figure 3.19: Plots of the constraint results with the system (3.1) under the controller (3.63).

3.4.4 Simulations

We then show the simulation result using the algorithm (3.63). Assume that agent j is assigned a constraint function $y_j^p(t) - x_j^p(t) - \cos(t) \leq 0$, for all $j \in [1, \cdots, 6]$, and agent k is assigned a constraint function $y_k^p(t) + x_k^p(t) - t - 3 = 0$, for all $k \in [7, \cdots, 12]$. The initial states $x_i^p(0),\ i \in \mathcal{V}$ are generated randomly from the range $[-10, 0]$, $y_i^p(0) = x_i^p(0) - 2,\ \forall i \in \mathcal{V}$. Therefore, the initial condition (3.30) is satisfied. We choose $\kappa = 12, \alpha = 15,\ \rho(t) = 10\exp(0.05t)$, and $z_i(0) = 0,\ s_i(0) = 5$ for all $i \in \mathcal{V}$. Therefore, the initial conditions (3.69) and (3.70) and the gain conditions (3.29) and (4.23) are satisfied.

The state trajectories of the agents are shown in Figure 3.18. We can see that all the agents converge to the vicinity of the optimal trajectory eventually which is consistent with Theorem 33. The constraint results are shown in Figure 3.19. In our simulation, agents $1-6$ are assigned the constraint function $y_i^p(t) - x_i^p(t) - \cos(t) \leq 0,\ i \in [1, \cdots, 6]$, and thus all $y_i^p(t) - x_i^p(t) - \cos(t) - 1/\rho(t),\ i \in [1, \cdots, 6]$ always remain negative. Moreover, all the equality constraint functions $y_i^p(t) + x_i^p(t) - t - 3,\ i \in [7, \cdots, 12]$ converge to the neighborhood of the zero line eventually.

3.5 Conclusions

In this Chapter, we have studied the distributed continuous-time constrained optimization problem with time-varying objective functions and time-varying constraints. The goal is for a set of networked agents to cooperatively track the time-varying optimal solution that minimizes the summation of all the local time-varying objective functions subject to all the local time-varying constraints, where each agent has only local information and local interaction. We have proposed distributed sliding-mode algorithms built on the Hessian-based optimization methodology. We have shown that asymptotical convergence to the optimal solution or its vicinity is guaranteed under some reasonable assumptions. Both numerical simulation results and experimental results are given to illustrate the theoretical algorithm.

Chapter 4

Distributed Average Tracking in Weight-Unbanlaced Networks

In this Chapter, we study distributed time-varying optimization under possibly weight-unbalanced directed networks—the most general and thus most challenging case from the network topology perspective. Particularly, we aim to seek distributed time-varying optimization algorithms for quadratic objective functions under unbalanced graphs, which is equivalent to distributed average tracking problem. In distributed average tracking problem, a collection of agents work collaboratively, subject to local communication, to track the average of a set of reference signals, each of which is available to a single agent. For this purpose, we propose a distributed algorithm based on a chain of two integrators which are coupled with a distributed estimator. It is found that the convergence depends on not only the network topology but also the deviations among the reference signal accelerations. Another primary interest of this note stems from the dynamics perspective—a point perceived as a main source of control design difficulty for multi-agent systems. Indeed, we devise a nonlinear algorithm which is capable of achieving distributed average tracking under weight-unbalanced directed networks for agents subject to high-order integrator dynamics. The results show that the convergence to the vicinity of the average of the reference signals is guaranteed as long as the signals' states and control inputs are all bounded. Both algorithms are robust to initialization errors, i.e., distributed average tracking is insured even if the agents are not correctly initialized, enabling the potential applications in a wider spectrum of application domains.

4.1 Preliminary

In this section, we show the connection of distributed average tracking with distributed time-varying optimization. For the distributed time-varying optimization problem given by (3.3), consider quadratic objective functions, i.e.,

$$f_i[x_i(t), t] = [x_i(t) - r_i(t)]^2, \tag{4.1}$$

where $r_i(t) \in \mathbb{R}^m$ is a private time-varying signal of agent i. The gradient of the quadratic objective function in (4.1) is $\nabla f_i[x_i(t), t] = 2[x_i(t) - r_i(t)]$. Then it follows from Lemma 7 that the optimal solution of problem (3.3) with quadratic objective function in (4.1) is given by

$$\begin{aligned} &x^*(t) = x_i(t) = x_j(t), \forall i, j \in \mathcal{V}, \\ &\sum_{i=1}^{n} 2[x^*(t) - r_i(t)] = \mathbf{0}_m. \end{aligned} \tag{4.2}$$

Then we have $x^*(t) = x_i(t) = x_j(t) = \frac{1}{n}\sum_{i=1}^{n} r_i(t)$. It is obvious that the optimal solution is the average of all the signals. Here, we call this problem distributed average tracking.

4.2 Distributed Average Tracking For Single-Integrator Dynamics

4.2.1 Problem Formulation

In this section, the distributed average tracking problem for multi-agent systems with single-integrator dynamics over weight-unbalanced directed graphs is studied. Consider a multi-agent system consisting of n agents with an interaction topology described by a weighted directed graph $\mathcal{G}$.

Assumption 12. *The directed graph $\mathcal{G}$ is time invariant and strongly connected* [1].

Suppose that the agents follow the single-integrator dynamics in (3.1). Each agent has a time-varying reference signal $r_i(t) \in \mathbb{R}^m, i = 1, ..., n$, satisfying

$$\dot{r}_i(t) = v_i^r(t), \qquad \dot{v}_i^r(t) = a_i^r(t), \tag{4.3}$$

where $v_i^r(t) \in \mathbb{R}^m$ and $a_i^r(t) \in \mathbb{R}^m$ are, respectively, the velocity and acceleration of the ith reference signal. For example, the reference signal r_i might be the position, sensed by the ith camera, of a mobile target of interest.

Our main objective is to design a distributed algorithm for agent $i \in \mathcal{V}$ based on $r_i(t)$, $v_i^r(t)$, $a_i^r(t)$, $x_i(t)$ and $x_j(t), j \in \mathcal{N}_i$, such that it tracks the average of all the time-varying reference signals, i.e.,

$$\lim_{t\to\infty} \left\| x_i(t) - (1/n) \sum_{j=1}^{n} r_j(t) \right\|_2 = 0. \tag{4.4}$$

[1] Note that there is no requirement that $\mathcal{G}$ be weight-balanced.

We call a distributed average tracking algorithm robust to initialization errors if the objective (4.4) can be achieved regardless of the agents' initial states. For notational simplicity, we will remove the time index t from variables in the reminder of the section.

4.2.2 Algorithm Design

We propose the following algorithm:

$$\begin{aligned}
u_i &= -\kappa(x_i - r_i) - \kappa z_{ii} \sum_{j \in \mathcal{N}_i} a_{ij}(x_i - x_j) + v_i^r - w_{1i}, \\
\dot{w}_{1i} &= \kappa^2 z_{ii} \sum_{j \in \mathcal{N}_i} a_{ij}(x_i - x_j) - z_{ii} \sum_{j \in \mathcal{N}_i} a_{ij}(w_{2i} - w_{2j}), \\
\dot{w}_{2i} &= w_{1i} - \kappa r_i - v_i^r, \quad \dot{z}_i = -\sum_{j \in \mathcal{N}_i} a_{ij}(z_i - z_j),
\end{aligned} \tag{4.5}$$

where $\kappa \in \mathbb{R}_{>0}$ is a positive control gain, $z_i \in \mathbb{R}^n$ is agent i's estimate of the left eigenvector corresponding to the zero eigenvalue of the Laplacian matrix, z_{ii} is the ith component of z_i, $w_{1i} \in \mathbb{R}^m$ and $w_{2i} \in \mathbb{R}^m$ are the internal states of a chain of two integrators, and a_{ij} is the (i,j)th entry of the adjacency matrix. We initialize the internal states w_{1i}, w_{2i}, and the estimators z_i to satisfy the following conditions:

$$\sum_{i=1}^{n} w_{1i}(0) = \mathbf{0}_m, \; z_{ij}(0) = 0, \; \forall i \neq j, \; z_{ii}(0) = 1, \; \forall i \in \mathcal{V}. \tag{4.6}$$

We note that each component of x_i is decoupled in (4.5). Therefore, in the following, we will only tackle the one-dimensional case, i.e., $m = 1$. The same conclusion holds for any $m \geq 2$ by using the Kronecker product. Substituting (4.5) into (3.1) leads to a vector form as

$$
\begin{aligned}
\dot{x} &= -\kappa(x-r) - \kappa Z_n \mathcal{L} x + v^r - w_1, \\
\dot{w}_1 &= \kappa^2 Z_n \mathcal{L} x - Z_n \mathcal{L} w_2, \\
\dot{w}_2 &= w_1 - \kappa r - v^r, \quad \dot{z} = -(\mathcal{L} \otimes I_n) z,
\end{aligned}
\tag{4.7}
$$

where $r = [r_1, ..., r_n]^T \in \mathbb{R}^n$, $v^r = [v_1^r, ..., v_n^r]^T \in \mathbb{R}^n$, $x = [x_1, ..., x_n]^T \in \mathbb{R}^n$, $w_1 = [w_{11}, ..., w_{1n}]^T \in \mathbb{R}^n$, $w_2 = [w_{21}, ..., w_{2n}]^T \in \mathbb{R}^n$, $z = [z_1^T, ..., z_n^T]^T \in \mathbb{R}^{n^2}$ and $Z_n = \text{diag}([z_{11}, z_{22}, ..., z_{nn}]) \in \mathbb{R}^{n \times n}$.

Lemma 34. *If Assumption 12 holds and $z(0)$ satisfies (4.6), then $\lim_{t\to\infty} Z_n \to P$, where P is defined in Lemma 2.*

Proof. We know that $z = \exp[(-\mathcal{L} \otimes I_n)t]z(0)$. By Lemma 2, it can be obtained that $\lim_{t\to\infty} z = \exp(\mathbf{1}_n p^T \otimes I_n) z(0) = \mathbf{1}_n \otimes p$ if $z(0)$ satisfies (4.6), yielding $\lim_{t\to\infty} Z_n \to P$. $\square$

Remark 35. *Compared with [83] which requires the network be undirected, the algorithm* (4.5) *can work for generic directed networks. Due to Lemma 34, we know that Z_n is utilized to estimate the matrix P. It follows from $\mathbf{1}_n^T P \mathcal{L} = \mathbf{0}_n^T$ that $P\mathcal{L}$ is equivalent to the Laplacian matrix of a balanced directed graph [63].*

Remark 36. *In the proposed algorithm* (4.5)*, a chain of two integrators with the internal states w_{1i} and w_{2i} are introduced to make* (4.5) *work for more general reference signals, the term $-\kappa(x_i - r_i)$ is introduced to achieve sum tracking, i.e.,* $\lim_{t\to\infty} \| \sum_{i=1}^n x_i - \sum_{i=1}^n r_i \|_2 = 0$, *and the term $-\kappa z_{ii} \sum_{j\in\mathcal{N}_i} a_{ij}(x_i - x_j)$ is introduced to achieve consensus with the aid of the chain of two integrators w_{1i} and w_{2i}. The distributed estimator given by the last equation in* (4.5) *is used by agent i to estimate the left eigenvector, corresponding to the zero eigenvalue, of the Laplacian matrix.*

Remark 37. *In the proposed algorithm (4.5), only correct initializations of internal states $w_{1i}(0)$ and $z_i(0)$ are needed, and correct initializations of agents' states $x_i(0)$ and $w_{2i}(0)$ are not required, which makes the algorithm robust to the state initialization errors. Note that the initialization condition* (4.6) *can be easily satisfied, e.g., to satisfy $\sum_{i=1}^{n} w_{1i}(0) = 0$, we can choose $w_{1i} = 0, \quad \forall i = 1, ..., n$.*

4.2.3 Algorithm Analysis

The main assumption and result of this section are stated in the following theorem.

Assumption 13. *The deviations among the accelerations of the references all tend to zero, i.e., $\lim_{t\to\infty}(a_i^r - a_j^r) = 0, i \neq j$.*

Theorem 38. *Using (4.5) for (3.1), if Assumptions 12 and 13 and the initial condition (4.6) hold, and $\kappa \gg 1$, then $\lim\limits_{t\to\infty} \left\| x_i - \frac{1}{n}\sum_{j=1}^{n} r_j \right\|_2 = 0$ for all $i = 1, ..., n$.*

Proof. Define $\tilde{x} = x - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T r$, $\tilde{w}_1 = w_1 - \kappa\Omega r - \Omega v^r$, $\tilde{w}_2 = Z_n\mathcal{L}w_2 + \kappa\Omega v^r + \Omega a^r$, and $Y = [\tilde{x}^T, \tilde{w}_1^T, \tilde{w}_2^T]^T$ where $\Omega = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$. It can be verified that $\mathcal{L}\tilde{x} = \mathcal{L}x$ and $\mathcal{L}\Omega = \mathcal{L}$. The first three equations in (4.7) can be rewritten in terms of Y as

$$\dot{Y} = f(Y) + g(Y) + h, \tag{4.8}$$

where

$$f(Y) = \begin{bmatrix} -\kappa\tilde{x} - \kappa P\mathcal{L}\tilde{x} - \tilde{w}_1 \\ \kappa^2 P\mathcal{L}\tilde{x} - \tilde{w}_2 \\ P\mathcal{L}\tilde{w}_1 \end{bmatrix}, g(Y) = \begin{bmatrix} \kappa(P - Z_n)\mathcal{L}\tilde{x} \\ \kappa^2(Z_n - P)\mathcal{L}\tilde{x} \\ (Z_n - P)\mathcal{L}\tilde{w}_1 \end{bmatrix}, h = \begin{bmatrix} 0 \\ 0 \\ \kappa\Omega a^r + \Omega\dot{a}^r \end{bmatrix}.$$

Based on Assumption 13, we know that h will approach zero as time goes to infinity. Therefore, by taking h in (4.8) as the system input, we first analyze the stability and convergence properties of the unforced system, i.e.,

$$\dot{Y} = f(Y) + g(Y), \tag{4.9}$$

Due to Lemma 34, we know that as $t \to \infty$, $Z_n - P$ tends to zero, so by Corollary 9.1 and Lemma 9.5 in [48] the convergence of (4.9) can be analyzed via $\dot{Y} = f(Y)$ only, i.e.,

$$\begin{bmatrix} \dot{\tilde{x}} \\ \dot{\tilde{w}}_1 \\ \dot{\tilde{w}}_2 \end{bmatrix} = \tilde{A} \begin{bmatrix} \tilde{x} \\ \tilde{w}_1 \\ \tilde{w}_2 \end{bmatrix}, \tag{4.10}$$

where

$$\tilde{A} = \begin{bmatrix} -\kappa - \kappa P\mathcal{L} & -I_n & \bar{\mathbf{0}}_n \\ \kappa^2 P\mathcal{L} & \bar{\mathbf{0}}_n & -I_n \\ \bar{\mathbf{0}}_n & P\mathcal{L} & \bar{\mathbf{0}}_n \end{bmatrix}.$$

In the following, we show that the dynamical system (4.10) is stable and convergent by studying the dynamics of two related systems. Define $T_1 = \begin{bmatrix} q_1 & Q^T \end{bmatrix}^T$, where $q_1 = \frac{1}{\sqrt{n}}\mathbf{1}_n$, $Qq_1 = \mathbf{0}_{n-1}$ and $QQ^T = I_{n-1}$. It follows that $T_1 P\mathcal{L} T_1^T = \begin{bmatrix} 0 & \mathbf{0}_{n-1}^T \\ \mathbf{0}_{n-1} & \Lambda \end{bmatrix}$, where Λ is an upper triangular matrix whose diagonal entries are the nonzero eigenvalues of $P\mathcal{L}$. Thus $\hat{x} = \begin{bmatrix} \hat{x}_1 & \hat{x}_{2:n}^T \end{bmatrix}^T = T_1\tilde{x}$, $\hat{w}_1 = \begin{bmatrix} \hat{w}_{11} & \hat{w}_{12:1n}^T \end{bmatrix}^T = T_1\tilde{w}_1$ and $\hat{w}_2 = \begin{bmatrix} \hat{w}_{21} & \hat{w}_{22:2n}^T \end{bmatrix}^T = T_1\tilde{w}_2$, where $\hat{x}_1, \hat{w}_{11}$ and $\hat{w}_{21} \in \mathbb{R}$. We can rewrite (4.10) as

$$\begin{bmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{w}}_{11} \\ \dot{\hat{w}}_{21} \end{bmatrix} = \underline{\mathrm{A}} \begin{bmatrix} \hat{x}_1 \\ \hat{w}_{11} \\ \hat{w}_{21} \end{bmatrix}, \quad \begin{bmatrix} \dot{\hat{x}}_{2:n} \\ \dot{\hat{w}}_{12:1n} \\ \dot{\hat{w}}_{22:2n} \end{bmatrix} = \hat{A} \begin{bmatrix} \hat{x}_{2:n} \\ \hat{w}_{12:1n} \\ \hat{w}_{22:2n} \end{bmatrix}, \tag{4.11}$$

where

$$\underline{A} = -\begin{bmatrix} \kappa & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \hat{A} = \begin{bmatrix} -\kappa - \kappa\Lambda & -I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \kappa^2\Lambda & \bar{\mathbf{0}}_{n-1} & -I_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \Lambda & \bar{\mathbf{0}}_{n-1} \end{bmatrix}.$$

The matrix $\underline{A}$ has two eigenvalues 0 (with multiplicity 2) and $-\kappa$. Define

$$T_2 = \begin{bmatrix} I_{n-1} & \bar{\mathbf{0}}_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \kappa I_{n-1} & I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \bar{\mathbf{0}}_{n-1} & I_{n-1} \end{bmatrix},$$

such that $[\bar{x}^T, \bar{w}_1^T, \bar{w}_2^T]^T = T_2[\hat{x}_{2:n}^T, \hat{w}_{12:1n}^T, \hat{w}_{22:2n}^T]^T$. It follows from Eq. (4.11) that

$$\begin{bmatrix} \dot{\bar{x}} \\ \dot{\bar{w}}_1 \\ \dot{\bar{w}}_2 \end{bmatrix} = \bar{A} \begin{bmatrix} \bar{x} \\ \bar{w}_1 \\ \bar{w}_2 \end{bmatrix}, \tag{4.12}$$

where

$$\bar{A} = \begin{bmatrix} -\kappa\Lambda & -I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & -\kappa I_{n-1} & -I_{n-1} \\ -\kappa\Lambda & \Lambda & \bar{\mathbf{0}}_{n-1} \end{bmatrix}.$$

The determinant $\det(\lambda I_{n-1} - \bar{A})$ is given by

$$\det \begin{bmatrix} \lambda I_{n-1} + \kappa\Lambda & I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \lambda I_{n-1} + \kappa I_{n-1} & I_{n-1} \\ \kappa\Lambda & -\Lambda & \lambda I_{n-1} \end{bmatrix} = \det \begin{bmatrix} \lambda I_{n-1} + \kappa\Lambda & I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \lambda I_{n-1} + \kappa I_{n-1} & I_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \bar{\mathbf{0}}_{n-1} & \Gamma \end{bmatrix}, \tag{4.13}$$

where $\Gamma = \lambda I_{n-1} + (\lambda I_{n-1} + \kappa I_{n-1})^{-1}[\Lambda + \kappa(\lambda I_{n-1} + \kappa\Lambda)^{-1}\Lambda]$. Noting that the inverse of an upper triangular matrix is also an upper triangular matrix, and the multiplication of two upper triangular matrices is also an upper triangular matrix, it follows that Γ is an upper triangular matrix. Define $\Xi \in \mathbb{R}^{n-1\times n-1}$ where $\Xi_{i,i} = \Lambda_{i,i}$ for all $i = 1, 2, ..., n-1$ and $\Xi_{i,j} = 0$ for all $i \neq j$. We have

$$\det \begin{bmatrix} \lambda I_{n-1} + \kappa\Lambda & I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \lambda I_{n-1} + \kappa I_{n-1} & I_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \bar{\mathbf{0}}_{n-1} & \Gamma \end{bmatrix} = \det \begin{bmatrix} \lambda I_{n-1} + \kappa\Xi & \bar{\mathbf{0}}_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \lambda I_{n-1} + \kappa I_{n-1} & \bar{\mathbf{0}}_{n-1} \\ \bar{\mathbf{0}}_{n-1} & \bar{\mathbf{0}}_{n-1} & \tilde{\Gamma} \end{bmatrix}, \tag{4.14}$$

where $\tilde{\Gamma} = \lambda I_{n-1} + (\lambda I_{n-1} + \kappa I_{n-1})^{-1}[\Xi + \kappa(\lambda I_{n-1} + \kappa\Xi)^{-1}\Xi]$. The eigenvalues λ of $\bar{A}$ can be obtained by calculating the following equations,

$$\lambda^3 + \kappa\lambda^2 + \kappa\Xi_{ii}\lambda^2 + \kappa^2\Xi_{ii}\lambda + \Xi_{ii}\lambda + \kappa\Xi_{ii}^2 + \kappa\Xi_{ii} = 0,\ i = 1, .., n-1. \tag{4.15}$$

To solve Eq. (4.15), we consider the corresponding perturbed cubic equation:

$$\lambda^3 + \left(\kappa + \kappa\Xi_{ii} + \frac{1}{\kappa} - \varepsilon\right)\lambda^2 + \left(\kappa^2\Xi_{ii} + \Xi_{ii} + 1 - \kappa\varepsilon\right)\lambda + \kappa\Xi_{ii}^2 + \kappa^2\Xi_{ii}\varepsilon = 0,\ i = 1, .., n-1. \tag{4.16}$$

With ε beging the perturbation parameter, it is worth noting that when $\varepsilon = \frac{1}{\kappa}$, the perturbed cubic equation (4.16) reduces to (4.15). It follows from $\kappa \gg 1$ that $(1/\kappa) \ll 1$. Based on perturbation theory [38], the eigenvalues λ are given by $\lambda = \lambda_0 + \varepsilon\lambda_1 + O(\varepsilon) = \lambda_0 + \frac{1}{\kappa}\lambda_1 + O(\varepsilon)$, where λ_0 is the solution of the following equations

$$\left(\lambda + \frac{1}{\kappa}\right)(\lambda + \kappa)(\lambda + \kappa\Xi_{ii}) = 0, \quad i = 1, .., n-1, \tag{4.17}$$

and $\lambda_1 = \frac{\lambda_0^2 + \kappa\lambda_0 - \kappa^2\Xi_{ii}^2}{\lambda_0^2 + 2\lambda_0 + 2\Xi_{ii}\kappa + 2\kappa + 2/\kappa + \Xi_{ii}\kappa^2 + \Xi_{ii} + 1}, \quad i = 1, ..., n-1$, with $O(\varepsilon)$ being the higher order term.

It follows from Eq. (4.17) that for each $i = 1, \cdots, n-1$, $\lambda_0 = -1/\kappa, -\kappa, -\kappa\Xi_{ii}$, along with

$$\lambda_1 = \begin{cases} \frac{(1/\kappa)^2 - 1 - \kappa^2\Xi_{ii}^2}{(1/\kappa)^2 + 2\Xi_{ii}\kappa + 2\kappa + \Xi_{ii}\kappa^2 + \Xi_{ii} + 1} & \text{if } \lambda_0 = -\frac{1}{\kappa}, \\ \frac{-\kappa^2\Xi_{ii}^2}{(\kappa)^2 + 2\Xi_{ii}\kappa + 2/\kappa + \Xi_{ii}\kappa^2 + \Xi_{ii} + 1} & \text{if } \lambda_0 = -\kappa, \\ \frac{-\kappa^2\Xi_{ii}}{(\kappa\Xi_{ii})^2 + 2\kappa + 2/\kappa + \Xi_{ii}\kappa^2 + \Xi_{ii} + 1} & \text{if } \lambda_0 = -\kappa\Xi_{ii}. \end{cases} \tag{4.18}$$

Recall that the nonzero eigenvalues of $P\mathcal{L}$ all have positive real parts, therefore λ_0 and λ_1 all have negative real parts, which indicates that the eigenvalues λ of (4.15) all have negative real parts. Hence the dynamical system (4.12) is exponentially stable. Noting that T_1 and T_2 are all nonsingular matrices, the dynamical system (4.10) must be exponentially stable.

The null space of the system matrix $\tilde{A}$ of (4.10) is spanned by $[\mathbf{1}_n^T, -\kappa\mathbf{1}_n^T, \mathbf{0}_n^T]^T$, the eigenvector associated with the zero eigenvalue. Therefore, (4.10) converges exponentially fast to the set $\{(\tilde{x}, \tilde{w}_1, \tilde{w}_2) \mid \tilde{x} = a\mathbf{1}_n, \tilde{w}_1 = -a\kappa\mathbf{1}_n, \tilde{w}_2 = \mathbf{0}_n, a \in \mathbb{R}\}$. According to the definition of $\tilde{w}_1$, we know that $\mathbf{1}_n^T\tilde{w}_1 = \mathbf{1}_n^T w_1$. It follows from $\mathbf{1}_n^T\left(\kappa^2 P\mathcal{L}x - P\mathcal{L}w_2\right) = 0$ that $\mathbf{1}_n^T\dot{w}_1 = 0$, leading to $\mathbf{1}_n^T\tilde{w}_1 = \mathbf{1}_n^T w_1 = \mathbf{1}_n^T w_1(0) = 0$. Therefore, if (4.6) holds, system (4.10) converges exponentially to the set $\{(\tilde{x}, \tilde{w}_1, \tilde{w}_2) \mid \tilde{x} = \mathbf{0}_n, \tilde{w}_1 = \mathbf{0}_n, \tilde{w}_2 = \mathbf{0}_n\}$. Based on Corollary 9.1 and Lemma 9.5 in [48], the perturbed system (4.9) is exponentially stable with respect to the equilibrium point. Therefore, under Assumption 13, Y in (4.8) asymptotically converges to $\mathbf{0}_{3n}$. □

In practice, the reference signals may not always satisfy Assumption 13. A more realistic assumption is to require the deviations among the reference signal accelerations be bounded, which is formally stated in the following.

Assumption 14. *The deviations among the accelerations of the reference signals are bounded, i.e., there exists* $\bar{a}^r \in \mathbb{R}_{>0}$ *such that* $\sup_{t\in[0,\infty)}(a_i^r - a_j^r) = \bar{a}^r, \forall i \in \mathcal{V}, j \in \mathcal{V}, i \neq j$.

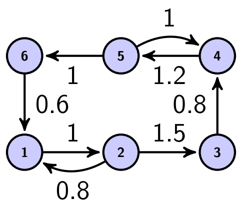

Figure 4.1: A weight-unbalanced directed communication topology.

Theorem 39. *Using* (4.5) *for* (3.1)*, if Assumption 14 and the internal state's initial conditions* (4.6) *hold, and* $\kappa \gg 1$*, then* $\sup\limits_{t\in[0,\infty)} \left\| x_i - \frac{1}{n}\sum\limits_{j=1}^{n} r_j \right\|_2$ *is bounded for all* $i = 1, ..., n$*.*

Proof. Because of the boundedness of h, system (4.8) is input-to-state stable by Lemma 4.6 given in [48]. If Assumption 14 holds, it follows from Definition 4.7 in [48] that the whole tracking error is upper bounded by $\lim\limits_{t\to\infty} \sup \|\tilde{x}(t)\|_2 \leq \lim\limits_{t\to\infty} \sup \|Y(t)\|_2 \leq \epsilon \bar{a}^r$, where ϵ is a positive constant. $\square$

4.2.4 Simulations

In this section, the proposed distributed average tracking algorithms are applied in the leader follower containment control problem (see Example: distributed formation control revisited in [51]) to reach on agreement at the geometric center of some leader robots so that the containment can be achieved by spreading the follower agents in a desired formation about this center. This kind of distributed containment control is particularly useful in some practical applications, such as cooperative protection of a group of robots through a polluted region. Therefore, we introduce an offset vector δ_i for each follower robot i and replace x_i in algorithm (4.5) with $x_i - \delta_i$. Here, $\delta_i - \delta_j$ defines the desired relative position from robot j to robot i in the formation. We consider $n = 6$, and use the weight-unbalanced directed graph in Figure 4.1 as the network topology. In this case, we implement the algorithm (4.5) to illustrate Theorem 38. Here the state of follower robot i is denoted by $x_i = [x_i^p, y_i^p]^T$, where (x_i^p, y_i^p) is the X-Y coordinates of follower robot i. We choose the following offset vectors: $\delta_1 = [2, 3.5]^T,\ \delta_2 = [0, 3.5]^T,\ \delta_3 = [-2, 3.5]^T,\ \delta_4 = [2, -3.5]^T,\ \delta_5 = [0, -3.5]^T,\ \delta_6 = [-2, -3.5]^T$. Each follower robot i has a corresponding leader robot whose state is the reference signal $r_i = [r_i^x, r_i^y]^T$, where (r_i^x, r_i^y) is the X-Y coordinates of leader robot i. The initial positions of the follower robots and leader robots are chosen randomly, and the initial velocities of the leader robots are all chosen as $v_i^r(0) = [1, 0]^T$. We choose the control gain κ as 15. We assume that leader robot i is operating in the environment with trajectories satisfying $\ddot{r}_i^x = 0$ and $\ddot{r}_i^y = \sin(t) + 0.03e^{-0.8t} i \sin(t)$.

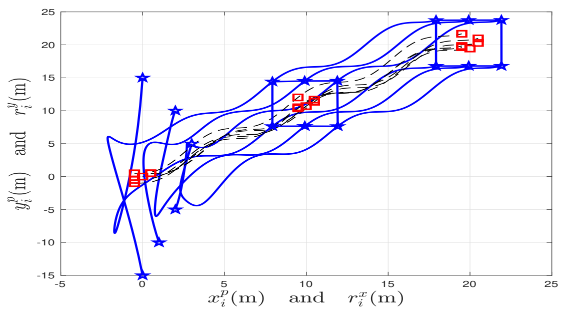

Figure 4.2: State trajectories of all the robots for the case in Section 4.2.4

The simulation results are shown in Figures 4.2-4.4. In particular, Figure 4.2 show the X-Y coordinates of all the robots, where the blue solid lines denote the positions of the follower robots and the black dashed lines denote the positions of the leader robots. Two snapshots at 10 s and 20s, denoted by the blue stars (follower robots) and red squares (leader robots) in Figure 4.2, show that all followers follow the group of all the leader robots in a containment fashion in both cases. Figure 4.3 (respectively, Figure 4.4) shows the state trajectories of all the follower robots without the offsets and the geometric center of all the leader robots in X-coordinate (respectively, in Y-coordinate). We can see that the follower robots track the geometric center of all leader robots eventually, which are consistent with Theorem 38.

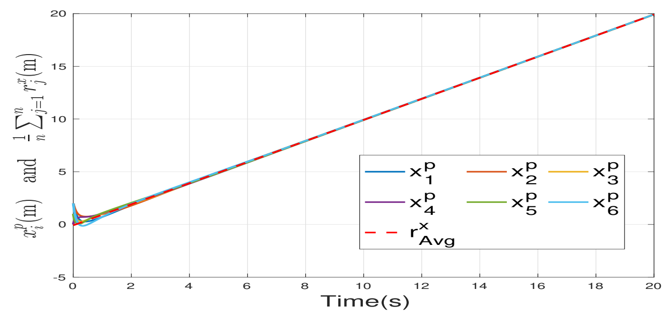

Figure 4.3: State trajectories of all the follower robots without the offsets and the geometric center of all the leader robots in X-coordinate for the case in Section 4.2.4

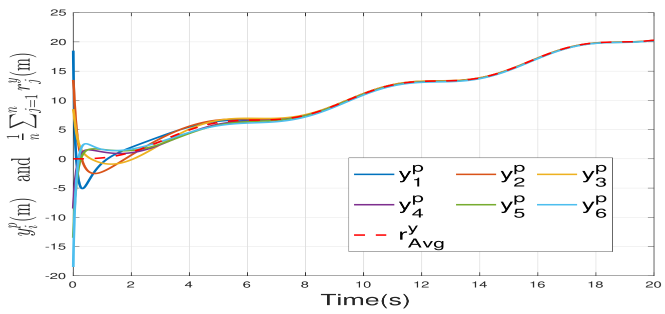

Figure 4.4: State trajectories of all the follower robots without the offsets and the geometric center of all the leader robots in Y-coordinate for the case in Section 4.2.4

4.3 Distributed Average Tracking For High-Order Integrator Dynamics

4.3.1 Problem Formulation

In this section, the distributed average tracking problem for multi-agent systems with high-order integrator dynamics over weight-unbalanced directed graphs is studied. In some applications, it might be more realistic to model the dynamics of the agents with high-order integrators. Unlike single-integrator dynamics, in the case of high-order integrator dynamics, the agents' system inputs might have a different dimension from that of the agents' state. Consider a network of n agents whose states are governed by

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t), \tag{4.19}$$

where $x_i(t) \in \mathbb{R}^m$, $u_i(t) \in \mathbb{R}$ are the system state and control input of the ith agent,[2] $A \in \mathbb{R}^{m\times m}$ is the state matrix, and $B \in \mathbb{R}^{m\times 1}$ is the input matrix. Here A and B are defined as

$$A = \begin{bmatrix} \mathbf{0}_{m-1} & I_{m-1} \\ 0 & \mathbf{0}_{m-1}^T \end{bmatrix}, \quad B = \begin{bmatrix} \mathbf{0}_{m-1} \\ 1 \end{bmatrix}.$$

Each agent has a time-varying reference signal $r_i(t) \in \mathbb{R}^m$ given by

$$\dot{r}_i(t) = Ar_i(t) + Bu_i^r(t), \tag{4.20}$$

where $r_i(t) \in \mathbb{R}^m$, $u_i^r(t) \in \mathbb{R}$ is the state and control input of the ith time-varying reference signal. The input $u_i^r(t)$ can be properly designed such that (4.20) can generate a general time-varying reference signal $r_i(t)$. The following standard assumption is made:

[2]Note that the dimensions of the control input u_i in Secs. III and IV are different.

Assumption 15. *The reference signals are bounded, and their control inputs are bounded, i.e., there exist $\bar{r} > 0$ and $\bar{u}^r > 0$ such that $\sup_{t\in[0,\infty)} \|r_i(t)\|_2 \leq \bar{r}$ and $\sup_{t\in[0,\infty)} \|u_i^r(t)\|_2 \leq \bar{u}^r$, for all $i \in \mathcal{V}$.*

Again, for notational simplicity, we will remove the time index t from variables in the reminder of this section and only keep it in some places when necessary.

4.3.2 Algorithm Design

We study the following control algorithm

$$\begin{aligned} u_i &= u_i^r + K_1(x_i - r_i) - \beta\tilde{h}\left[z_{ii}\sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right], \\ \dot{z}_i &= -\sum_{j\in\mathcal{N}_i} a_{ij}(z_i - z_j), \end{aligned} \tag{4.21}$$

where $z_i \in \mathbb{R}^n$ is agent i's estimate of the left eigenvector corresponding to the zero eigenvalue of the Laplacian matrix, z_{ii} is the ith element in vector z_i, a_{ij} is the (i,j)th element of the adjacency matrix $\mathcal{A}$, and $\tilde{h}(\cdot)$ is a function defined component-wise as $\tilde{h}(s) = \begin{cases} \frac{s}{|s|} & \text{if } |s| \geq \epsilon, \\ \frac{s}{\epsilon} & \text{otherwise,} \end{cases}$ where $\epsilon \in \mathbb{R}_{>0}$ is a small positive constant. In addition,

$$\begin{aligned} K_1 &\triangleq -(K_2A + K_2) \in \mathbb{R}^{1\times m}, \\ K_2 &\triangleq [C_{m-1}^0, C_{m-1}^1, \cdots, C_{m-1}^{m-1}] \in \mathbb{R}^{1\times m}, \end{aligned} \tag{4.22}$$

where $C_{m-1}^k = \frac{(m-1)!}{(m-1-k)!k!}$, $k \in [0, m-1]$, and β is a control gain satisfying

$$\beta > \frac{2n^{\frac{5}{2}}\sigma_{\max}(P\mathcal{L})[\bar{u}^r + \sigma_{\max}(K_1)\bar{r}]}{\lambda_2(\bar{\mathcal{L}})}, \tag{4.23}$$

where $\bar{\mathcal{L}}$ and P are defined in Lemma 2. We initialize the estimators $z_i, i \in \mathcal{V}$, to satisfy the last two equations in (4.6). Define $z \in \mathbb{R}^{n^2} = [z_1^T, \cdots, z_n^T]^T$, $Z_n = \text{diag}([z_{11}, z_{22}, \cdots, z_{nn}]) \in$

$\mathbb{R}^{n\times n}$, $x \triangleq \left[x_1^T, \cdots, x_n^T\right]^T \in \mathbb{R}^{nm}$, $u \triangleq [u_1, \cdots, u_n]^T \in \mathbb{R}^n$, $r \triangleq [r_1^T, \cdots, r_n^T]^T \in \mathbb{R}^{nm}$ and $u^r \triangleq [u_1^r, \cdots, u_n^r]^T \in \mathbb{R}^n$. Then the closed-loop system (4.19) can be written in a vector form as

$$\begin{aligned}\dot{x} &= (I_n \otimes A)x + (I_n \otimes B)\{u^r + (I_n \otimes K_1)(x - r) - \beta\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]\},\\ \dot{z} &= -(\mathcal{L} \otimes I_n)z.\end{aligned} \tag{4.24}$$

Remark 40. *For the high-order integrator case in Section IV, we only tackle the one-dimension case (i.e., $x_i \in \mathbb{R}^m$ and $u_i \in \mathbb{R}$ for m-order integrators), since each dimension of the high-order integrators is decoupled. Please note that our alrogithm can be easily extended to multi-dimension high-order integrator cases (i.e., $x_i \in \mathbb{R}^{mq}$ and $u_i \in \mathbb{R}^q$ for m-order integrators) by using the Kronecker product.*

Remark 41. *In the proposed algorithm* (4.21)*, the term $K_1(x_i - r_i)$ is introduced to achieve sum tracking (i.e., $\lim_{t\to\infty} \|\sum_{i=1}^n x_i(t) - \sum_{i=1}^n r_i(t)\|_2 = 0$) with the help of the distributed estimator z_i, and the term $-\beta\tilde{h}\left[z_{ii}\sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right]$ is introduced to guarantee consensus.*

4.3.3 Algorithm Analysis

This subsection establishes the convergence properties of the system (4.19) under the controller (4.21).

Lemma 42. *For any strongly connected directed graph $\mathcal{G}$ of order n, let $(P\mathcal{L})^+ \in \mathbb{R}^{n\times n}$ be the generalized inverse of $P\mathcal{L}$ with P and $\mathcal{L}$ being defined in Section II.B, we have $(P\mathcal{L})^+(P\mathcal{L}) = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$.*

Proof. Note that $\mathbf{1}_n^T P\mathcal{L} = \mathbf{0}_n^T$ and $P\mathcal{L}\mathbf{1}_n = \mathbf{0}_n$, i.e., $P\mathcal{L}$ can be viewed as the Laplacian matrix of a weight-balanced directed graph. Consequently, the proof follows directly from the proof of Lemma 3 in [33]. $\square$

Lemma 43. *Let Assumption 12 hold. Using* (4.21) *for* (4.19)*, if* $\|(P\mathcal{L} \otimes K_2)x(t)\|_1$ *is bounded for all* $t \geq 0$ *and*

$$\lim_{t\to\infty} \sup \|(P\mathcal{L} \otimes K_2)x(t)\|_1 \leq \bar{b}, \tag{4.25}$$

where $\bar{b}$ *is an arbitrary positive constant, then* $\lim\limits_{t\to\infty} \sup \|(\Omega \otimes I_m)x(t)\|_1 \leq \|(P\mathcal{L})^+ \otimes I_m\|_1 2^{m-1}\bar{b}\sum_{i=1}^m i!\left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2 \rfloor}\right)$*, where* $\Omega = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$*, and* $(P\mathcal{L})^+$ *is the generalized inverse of* $P\mathcal{L}$*.*

Proof. Define $X = [x_1, \cdots, x_n] \in \mathbb{R}^{m\times n}$, where $x_i \in \mathbb{R}^m,\ i \in [1,n]$, is defined in (4.19). Define $X_i \in \mathbb{R}^{1\times n}$ as the ith row of X. It follows from Proposition 1 and (4.22) that

$$\begin{aligned}&(P\mathcal{L} \otimes K_2)x = (P\mathcal{L} \otimes K_2)\text{vec}(X) = \text{vec}[K_2 X (P\mathcal{L})^T]\\ =&\text{vec}[(C_{m-1}^0 X_1 + \cdots + C_{m-1}^{m-1} X_m)(P\mathcal{L})^T]\\ =&C_{m-1}^0 \text{vec}[X_1(P\mathcal{L})^T] + \cdots + C_{m-1}^{m-1}\text{vec}[X_m(P\mathcal{L})^T]\\ =&C_{m-1}^0 \tilde{X}_1 + \cdots + C_{m-1}^{m-1}\tilde{X}_m,\end{aligned} \tag{4.26}$$

where $\tilde{X}_i = \text{vec}[X_i(P\mathcal{L})^T]$. For $\ell \in [0, m-1]$, define

$$s_\ell = C_{m-\ell-1}^0 \tilde{X}_1 + C_{m-\ell-1}^1 \tilde{X}_2 + \cdots + C_{m-\ell-1}^{m-\ell-1}\tilde{X}_{m-\ell}, \tag{4.27}$$

and thus

$$\dot{s}_\ell = C_{m-\ell-1}^0 \dot{\tilde{X}}_1 + C_{m-\ell-1}^1 \dot{\tilde{X}}_2 + \cdots + C_{m-\ell-1}^{m-\ell-1}\dot{\tilde{X}}_{m-\ell}.$$

By (4.19), we have $\dot{\tilde{X}}_k = \tilde{X}_{k+1}$ for $k \in [1, m-1]$. It follows that for $\ell \in [1, m-1]$,

$$\begin{aligned} s_\ell + \dot{s}_\ell &= C^0_{m-\ell-1}\tilde{X}_1 + C^1_{m-\ell-1}\tilde{X}_2 + \cdots + C^{m-\ell-1}_{m-\ell-1}\tilde{X}_{m-\ell} + C^0_{m-\ell-1}\dot{\tilde{X}}_1 + C^1_{m-\ell-1}\dot{\tilde{X}}_2 \\ &\qquad + \cdots + C^{m-\ell-1}_{m-\ell-1}\dot{\tilde{X}}_{m-\ell} \\ &= C^0_{m-\ell-1}\tilde{X}_1 + C^1_{m-\ell-1}\tilde{X}_2 + \cdots + C^{m-\ell-1}_{m-\ell-1}\tilde{X}_{m-\ell} + C^0_{m-\ell-1}\tilde{X}_2 + C^1_{m-\ell-1}\tilde{X}_3 \\ &\qquad + \cdots + C^{m-\ell-1}_{m-\ell-1}\tilde{X}_{m-\ell+1}. \end{aligned}$$

Because $C^k_n = C^{k-1}_{n-1} + C^k_{n-1}$ and $C^0_n = C^n_n = 1$, we have for $\ell \in [1, m-1]$,

$$\begin{aligned} s_\ell + \dot{s}_\ell &= C^0_{m-\ell-1}\tilde{X}_1 + \big(C^1_{m-\ell-1} + C^0_{m-\ell-1}\big)\tilde{X}_2 + \cdots \\ &\qquad + \big(C^{m-\ell-1}_{m-\ell-1} + C^{m-\ell-2}_{m-\ell-1}\big)\tilde{X}_{m-\ell} + C^{m-\ell-1}_{m-\ell-1}\tilde{X}_{m-\ell+1} \\ &= C^0_{m-\ell}\tilde{X}_1 + C^1_{m-\ell}\tilde{X}_2 + \cdots \\ &\qquad + C^{m-\ell-1}_{m-\ell}\tilde{X}_{m-\ell} + C^{m-\ell}_{m-\ell}\tilde{X}_{m-\ell+1} = s_{\ell-1}. \end{aligned} \tag{4.28}$$

The proof will proceed by the mathematical induction method. Recall the definition of s_ℓ given by (4.27). It follows from (4.26) that $s_0 = C^0_{m-1}\tilde{X}_1 + \cdots + C^{m-1}_{m-1}\tilde{X}_m = (P\mathcal{L} \otimes K_2)x$. Therefore, it follows from (4.25) that $s_0(t)$ is bounded for all $t \geq 0$ and $\lim\sup_{t\to\infty} \|s_0(t)\|_1 \leq \bar{b}$. Next, we will prove that for $\ell \in [1, m-1]$, if $s_{\ell-1}(t)$ is bounded for all $t \geq 0$ and $\lim\sup_{t\to\infty} \|s_{\ell-1}(t)\|_1 \leq \tilde{b}$ with $\tilde{b}$ being a positive constant, then $s_\ell(t)$ is bounded for all $t \geq 0$ and $\lim\sup_{t\to\infty} \|s_\ell(t)\|_1 \leq 2\tilde{b}$, using the input-to-state stability concept. It follows from (4.28) that $\dot{s}_\ell = -s_\ell + s_{\ell-1}$, which is obviously an input-to-state stable system by viewing s_ℓ as the state and $s_{\ell-1}$ as the input. Consider the Lyapunov function

$$\bar{V}(s_\ell) = \frac{1}{2}\|s_\ell\|_2^2.$$

It follows that $\alpha_1(\|s_\ell\|_2) \leq \bar{V}(s_\ell) \leq \alpha_2(\|s_\ell\|_2)$, where $\alpha_1(y) = \frac{1}{2}y^2$ and $\alpha_2(y) = \frac{1}{2}y^2$ are both class $\mathcal{K}_\infty$ functions. The derivative of $\bar{V}(s_\ell)$ is given by

$$\dot{\bar{V}}(s_\ell) = s_\ell^T \dot{s}_\ell = s_\ell^T(-s_\ell + s_{\ell-1}) \leq -\|s_\ell\|_2^2 + \|s_\ell\|_2\|s_{\ell-1}\|_2.$$

It follows that for all $\|s_\ell\|_2 \geq \rho(\|s_{\ell-1}\|_2)$, where $\rho(y) = 2y$ is a class $\mathcal{K}$ function, $\dot{\bar{V}}(s_\ell) \leq -\frac{1}{2}\|s_\ell\|_2^2$. Based on Theorem 4.19 and Definition 4.7 in [48], the system $\dot{s}_\ell = -s_\ell + s_{\ell-1}$ is an input-to-state stable system by viewing s_ℓ as the state and $s_{\ell-1}$ as the input, and there must exist a class $\mathcal{KL}$ function $\alpha(\cdot, t)$ and a class $\mathcal{K}$ function $\gamma(y) = \alpha_1^{-1} \circ \alpha_2 \circ \rho = 2y$ such that for any initial state $s_\ell(t_0)$ and any bounded input $s_{\ell-1}(t)$, the solution $s_\ell(t)$ exists and satisfies

$$\|s_\ell(t)\|_1 \leq \alpha(\|s_\ell(t_0)\|_1, t - t_0) + 2\left(\sup_{t_0 \leq \tau \leq t} \|s_{\ell-1}(\tau)\|_1\right). \tag{4.29}$$

Since $\|s_{\ell-1}(t)\|_1 \leq \tilde{b}$ as $t \to \infty$, for an arbitrary number $\zeta > 0$, there must exist a time $T_1 > 0$ such that $\|s_{\ell-1}(t)\|_1 \leq \tilde{b} + \zeta$ for all $t \geq T_1$. Consequently, for all $t \geq T_1$, we have

$$\begin{aligned} \|s_\ell(t)\|_1 &\leq \alpha(\|s_\ell(T_1)\|_1, t - T_1) + 2\left(\sup_{T_1 \leq \tau \leq t} \|s_{\ell-1}(\tau)\|_1\right) \\ &\leq \alpha(\|s_\ell(T_1)\|_1, t - T_1) + 2\big(\tilde{b} + \zeta\big). \end{aligned} \tag{4.30}$$

Since $\alpha(\|s_\ell(T_1)\|_1, t - T_1)$ is a class $\mathcal{KL}$ function, $\alpha(\|s_\ell(T_1)\|_1, t - T_1) \to 0$ as $t \to \infty$. Therefore, there must exist a time $T_2 > T_1$ such that $\alpha(\|s_\ell(T_1)\|_1, t - T_1) < \zeta$ for all $t \geq T_2$. It follows that for all $t \geq T_2$, $\|s_\ell(t)\|_1 < \zeta + 2\tilde{b} + 2\zeta = 3\zeta + 2\tilde{b}$. Since ζ is an arbitrary positive number, we have $\|s_\ell(t)\|_1 \leq 2\tilde{b}$ as $t \to \infty$. Therefore, we obtain the conclusion that for $\ell \in [1, m-1]$, if $s_{\ell-1}(t)$ is bounded for all $t \geq 0$ and $\lim_{t\to\infty} \sup \|s_{\ell-1}(t)\| \leq \tilde{b}$, then $s_\ell(t)$ is bounded for all $t \geq 0$ and $\lim_{t\to\infty} \sup \|s_\ell(t)\| \leq 2\tilde{b}$. Since we have proved that $s_0(t)$ is bounded for all $t \geq 0$ and $\lim_{t\to\infty} \sup \|s_0(t)\|_1 \leq \bar{b}$, we have that $s_\ell(t), \ell \in [0, m-1]$, is bounded for all $t \geq 0$ and

$$\lim_{t\to\infty} \sup \|s_\ell(t)\|_1 \leq 2^\ell \bar{b}. \tag{4.31}$$

Then, we will derive the bound of $\|\tilde{X}_i\|_1,\ i \in [1,m]$, based on the bound of $\|s_\ell\|_1, \ell \in [0, m-1]$, using the mathematical induction method again. It can be verified that $s_{m-1} = \tilde{X}_1$ by (4.27). Then it follows from (4.31) that $\lim_{t\to\infty}\sup \|\tilde{X}_1(t)\|_1 = \lim_{t\to\infty}\sup \|s_{m-1}(t)\|_1 \le 2^{m-1}\bar{b}$. Therefore there exists a positive constant $\bar{B}_2 = 2^{m-1}\bar{b} > 2^{m-2}\bar{b}$ such that $\lim_{t\to\infty}\sup \|\tilde{X}_1(t)\|_1 \le \bar{B}_2$. Next, we will prove that for $i \in [2,m]$, if there exists a positive constant $\bar{B}_i \ge 2^{m-i}\bar{b}$ such that $\lim_{t\to\infty}\sup \|\tilde{X}_k(t)\|_1 \le \bar{B}_i,\ \forall k \in [1, i-1]$, then there exists a positive constant $\bar{B}_{i+1} = iC_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i \ge 2^{m-i-1}\bar{b}$ such that $\lim_{t\to\infty}\sup \|\tilde{X}_k(t)\|_1 \le \bar{B}_{i+1},\ \forall k \in [1,i]$. For $i \in [2,m]$, if there exists a positive constant $\bar{B}_i \ge 2^{m-i}\bar{b}$ such that $\lim_{t\to\infty}\sup \|\tilde{X}_k(t)\|_1 \le \bar{B}_i,\ \forall k \in [1,i-1]$, it follows from (4.27) and (4.31) that

$$\begin{aligned}
\lim_{t\to\infty}\sup \|\tilde{X}_i(t)\|_1 =& \lim_{t\to\infty}\sup \|s_{m-i}(t) - C_{i-1}^0\tilde{X}_1(t) - \cdots - C_{i-1}^{i-2}\tilde{X}_{i-1}(t)\|_1 \\
\le& \lim_{t\to\infty}\sup \|s_{m-i}(t)\|_1 + C_{i-1}^0 \lim_{t\to\infty}\sup \|\tilde{X}_1(t)\|_1 + \cdots \\
& + C_{i-1}^{i-2} \lim_{t\to\infty}\sup \|\tilde{X}_{i-1}(t)\|_1 \\
\le& C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor} \lim_{t\to\infty}\sup \|s_{m-i}(t)\|_1 + C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor} \lim_{t\to\infty}\sup \|\tilde{X}_1(t)\|_1 + \cdots \\
& + C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor} \lim_{t\to\infty}\sup \|\tilde{X}_{i-1}(t)\|_1 \\
\le& C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor} 2^{m-i}\bar{b} + C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i + \cdots + C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i \\
\le& C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i + C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i + \cdots + C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i = iC_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i,
\end{aligned}$$

where the last third inequality holds since $C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor} \ge 1$ is the biggest coefficient among all $C_{i-1}^l, l \in [0, i-1]$. Since $i > 1$ and $C_{i-1}^{\lfloor\frac{i-1}{2}\rfloor} \ge 1$, we have $iC_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i \ge \bar{B}_i$. If $\bar{B}_i \ge 2^{m-i}\bar{b}$, it follows that $iC_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i \ge 2^{m-i}\bar{b} \ge 2^{m-i-1}\bar{b}$. Therefore, for $i \in [2,m]$, if there exists a positive constant $\bar{B}_i \ge 2^{m-i}\bar{b}$ such that $\lim_{t\to\infty}\sup \|\tilde{X}_k(t)\|_1 \le \bar{B}_i,\ \forall k \in [1,i-1]$, then there exists a positive contant $\bar{B}_{i+1} = iC_{i-1}^{\lfloor\frac{i-1}{2}\rfloor}\bar{B}_i \ge 2^{m-i-1}\bar{b}$ such that $\lim_{t\to\infty}\sup \|\tilde{X}_k(t)\|_1 \le \bar{B}_{i+1},\ \forall k \in$

$[1, i]$. We have proved that there exists $\bar{B}_2 = 2^{m-1}\bar{b} > 2^{m-2}\bar{b}$ such that $\lim_{t\to\infty} \sup \|\tilde{X}_k(t)\|_1 \leq \bar{B}_2$, $\forall k \in [1]$. It thus follows that $\lim_{t\to\infty} \sup \|\tilde{X}_i(t)\|_1 \leq i! \left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2 \rfloor}\right) 2^{m-1}\bar{b}$ for $i \in [1, m]$.

Recall the definition of $\tilde{X}_i$. Based on Proposition 1, we have

$$\begin{aligned}\|(P\mathcal{L} \otimes I_m) \operatorname{vec}(X)\|_1 &= \left\|\operatorname{vec}\left[X(P\mathcal{L})^T\right]\right\|_1 \\ &= \sum_{i=1}^{m} \left\|\operatorname{vec}\left[X_i(P\mathcal{L})^T\right]\right\|_1 = \sum_{i=1}^{m} \left\|\tilde{X}_i\right\|_1.\end{aligned}$$

Therefore,

$$\begin{aligned}\lim_{t\to\infty} \sup \|(P\mathcal{L} \otimes I_m)\, x(t)\|_1 &= \lim_{t\to\infty} \sup \|(P\mathcal{L} \otimes I_m) \operatorname{vec}[X(t)]\|_1 \\ &= \lim_{t\to\infty} \sup \sum_{i=1}^{m} \left\|\tilde{X}_i(t)\right\|_1 \\ &\leq 2^{m-1}\bar{b} \sum_{i=1}^{m} i! \left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2 \rfloor}\right).\end{aligned}$$

It follows from Lemma 42 that if Assumption 12 holds, then $(P\mathcal{L})^+ P\mathcal{L} = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$, which further leads to

$$\begin{aligned}&\lim_{t\to\infty} \sup \|(\Omega \otimes I_m)\, x(t)\|_1 = \lim_{t\to\infty} \sup \left\|\left[(P\mathcal{L})^+ \otimes I_m\right](P\mathcal{L} \otimes I_m)\right. \\ &\left.\times\, x(t)\right\|_1 \leq \|(P\mathcal{L})^+ \otimes I_m\|_1 2^{m-1}\bar{b} \sum_{i=1}^{m} i! \left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2 \rfloor}\right).\end{aligned}$$

□

The main result of this section is given as follows.

Theorem 44. *Using* (4.21) *for* (4.19)*, if Assumptions 12 and 15, the gain condition* (4.23) *and last two initial conditions in* (4.6) *hold, then* $\lim_{t\to\infty} \sup \sum_{k=1}^{n} \left\|x_k(t) - (1/n)\sum_{j=1}^{n} r_j(t)\right\|_1 \leq \|(P\mathcal{L})^+ \otimes I_m\|_1 n\epsilon 2^{m-1} \sum_{i=1}^{m} i! \left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2 \rfloor}\right)$*, where* $(P\mathcal{L})^+$ *is the generalized inverse of matrix* $P\mathcal{L}$.

Proof. Consider the Lyapunov function candidate

$$V(x) = \sum_{i=1}^{n} \tilde{g}\left[p_i \sum_{j\in\mathcal{N}_i} a_{ij} K_2(x_i - x_j)\right], \tag{4.32}$$

where p_i is defined in Lemma 2, $\tilde{g}(s) : \mathbb{R} \to \mathbb{R}$ is a function defined as $\tilde{g}(s) = \begin{cases} |s| & \text{if } |s| \geq \epsilon, \\ \frac{s^2}{2\epsilon} + \frac{\epsilon}{2} & \text{otherwise.} \end{cases}$

It can be shown that the derivative of $V(x)$ is

$$\dot{V}(x) = \sum_{i=1}^{n} \xi_i \left[p_i \sum_{j\in\mathcal{N}_i} a_{ij} K_2(\dot{x}_i - \dot{x}_j)\right], \tag{4.33}$$

where $\xi_i \in \mathbb{R},\ i \in [1, \cdots, n]$, is defined as

$$\xi_i = \begin{cases} 1 & \text{if } p_i \sum_{j\in\mathcal{N}_i} a_{ij} K_2(x_i - x_j) \geq \epsilon, \\ -1 & \text{if } p_i \sum_{j\in\mathcal{N}_i} a_{ij} K_2(x_i - x_j) \leq -\epsilon, \\ \frac{p_i \sum_{j\in\mathcal{N}_i} a_{ij} K_2(x_i - x_j)}{\epsilon} & \text{otherwise.} \end{cases} \tag{4.34}$$

Define $\xi = [\xi_1, \cdots, \xi_n]^T \in \mathbb{R}^n$. Then the derivative of $V(x)$ along the trajectory of (4.24) can be calculated as follows:

$$\begin{aligned} \dot{V}(x) &= \xi^T (P\mathcal{L} \otimes K_2)\, \dot{x} \\ &= \xi^T (P\mathcal{L} \otimes K_2) \left((I_n \otimes A)x + (I_n \otimes B)\right. \\ &\qquad \left. \times \{u^r + (I_n \otimes K_1)(x - r) - \beta\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]\}\right) \\ &= \xi^T (P\mathcal{L} \otimes K_2 B)u^r + \xi^T[P\mathcal{L} \otimes K_2(A + BK_1)]x \\ &\quad - \xi^T (P\mathcal{L} \otimes K_2 B K_1)r - \beta\xi^T(P\mathcal{L} \otimes K_2 B)\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]. \end{aligned} \tag{4.35}$$

Recall the difinition of K_2 and B, we have $K_2 B = C_{m-1}^{m-1} = 1$, it follows that

$$\begin{aligned} \dot{V}(x) &= \xi^T P\mathcal{L}u^r + \xi^T[P\mathcal{L} \otimes K_2(A + BK_1)]x \\ &\quad - \xi^T(P\mathcal{L} \otimes K_1)r - \beta\xi^T P\mathcal{L}\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]. \end{aligned} \tag{4.36}$$

Consider the first term in (4.36). It follows from the Schwartz inequality that

$$\xi^T P\mathcal{L}u^r \leq \|\xi\|_2\|P\mathcal{L}\|_2\|u^r\|_2 \leq n^{\frac{3}{2}}\sigma_{\max}(P\mathcal{L})\bar{u}^r. \tag{4.37}$$

Similarly, we have $\xi^T(P\mathcal{L}\otimes K_1)r \leq n^{\frac{3}{2}}\sigma_{\max}(P\mathcal{L})\sigma_{\max}(K_1)\bar{r}$. Next, consider the second term in (4.36). Due to the definitions of K_1 and K_2 in (4.22), we know that $K_2(A+BK_1) = -K_2$. Recall the definition of ξ in (4.34), it follows that $\xi = \tilde{h}[(P\mathcal{L}\otimes K_2)x]$ and thus ξ_i and the ith element of $(P\mathcal{L}\otimes K_2)x$ have the same sign. Therefore $\xi^T[P\mathcal{L}\otimes K_2(A+BK_1)]x = -\xi^T(P\mathcal{L}\otimes K_2)x \leq 0$.

Consider the forth term in (4.36). Define $\eta \in \mathbb{R}^n = \tilde{h}[(Z_n\mathcal{L}\otimes K_2)x]$ and $\psi \in \mathbb{R}^n = \eta - \xi$, then we have

$$\begin{aligned}
&-\beta\xi^T P\mathcal{L}\tilde{h}[(Z_n\mathcal{L}\otimes K_2)x] = -\beta\xi^T P\mathcal{L}\eta \\
=&-\beta\xi^T P\mathcal{L}(\xi+\psi) = -\frac{\beta}{2}\xi^T P\mathcal{L}\xi - \frac{\beta}{2}\xi^T\mathcal{L}^T P\xi - \beta\xi^T P\mathcal{L}\psi \\
=&-\frac{\beta}{2}\xi^T\bar{\mathcal{L}}\xi - \beta\xi^T P\mathcal{L}\psi \leq -\frac{\beta}{2}\xi^T\bar{\mathcal{L}}\xi + \beta n^{\frac{1}{2}}\sigma_{\max}(P\mathcal{L})\|\psi\|_2.
\end{aligned}$$

Here $\bar{\mathcal{L}}$ is defined in Lemma 2. In addition, if there exists $i \in \mathcal{V}$ such that $\left|p_i\sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \geq \epsilon$, there is at least a 1 and a negative number or a -1 and a positive number in the elements of vector ξ. Therefore, there must exist a positive vector a such that $a^T\xi = 0$. It follows from Lemma 2 that if there exists $i \in \mathcal{V}$ such that $\left|p_i\sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \geq \epsilon$, then $\|\xi\|_2^2 \geq 1$ and

$$\tfrac{\beta}{2}\xi^T\bar{\mathcal{L}}\xi > \tfrac{\beta}{2n}\lambda_2(\bar{\mathcal{L}})\|\xi\|_2^2 \geq \tfrac{\beta}{2n}\lambda_2(\bar{\mathcal{L}}). \tag{4.38}$$

Then we have if there exists $i \in \mathcal{V}$ such that $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \geq \epsilon$,

$$\begin{aligned}\dot{V}(x) &\leq -\xi^T\big(P\mathcal{L}\otimes K_2\big)x + n^{\frac{3}{2}}\sigma_{\max}(P\mathcal{L})[\bar{u}^r + \sigma_{\max}(K_1)\bar{r}] \\ &\quad - \frac{\beta}{2n}\lambda_2(\bar{\mathcal{L}}) + \beta n^{\frac{1}{2}}\sigma_{\max}(P\mathcal{L})\|\psi\|_2 \\ &< -\xi^T\big(P\mathcal{L}\otimes K_2\big)x + \beta n^{\frac{1}{2}}\sigma_{\max}(P\mathcal{L})\|\psi\|_2.\end{aligned} \tag{4.39}$$

Here the last inequality is under the gain condition (4.23). Next, we show that all $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right|$ remains in a bounded region. According to the definition of $V(x)$, we know that the existence of $i \in \mathcal{V}$ such that $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \to \infty$ is a necessary and sufficient condition of $V(x) \to \infty$. If there exists $i \in \mathcal{V}$ such that $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \to \infty$, it will also hold that $\xi^T\big(P\mathcal{L}\otimes K_2\big)x \to \infty$ and $\dot{V}(x) < 0$, which will result in a bounded $V(x)$ and thus all bounded $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right|$. It follows that all $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right|$ and thus $V(x)$ remain bounded. According to the gain condition in (4.23), we know that there exists a positive number ς such that $\dot{V} \leq \varsigma - \xi^T\big(P\mathcal{L}\otimes K_2\big)x + \beta n^{\frac{1}{2}}\sigma_{\max}(P\mathcal{L})\|\psi\|_2$ if there exists $i \in \mathcal{V}$ such that $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \geq \epsilon$. It follows from Lemma 34 that $\psi \to \mathbf{0}_n$ as $t \to \infty$, therefore there must exist a time T_3 such that $\beta n^{\frac{1}{2}}\sigma_{\max}(P\mathcal{L})\|\psi\|_2 < \varsigma$ for all $t \geq T_3$. Then we have $\dot{V}(x) < 0$ for all $t \geq T_3$ if there exists $i \in \mathcal{V}$ such that $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \geq \epsilon$. Then we can get the conclusion that $\|(P\mathcal{L}\otimes K_2)x(t)\|_1$ is bounded for all $t \geq 0$, and all $\left|p_i \sum_{j\in\mathcal{N}_i} a_{ij}K_2(x_i - x_j)\right| \leq \epsilon$ as $t \to \infty$ and thus

$$\lim_{t\to\infty}\sup \|(P\mathcal{L}\otimes K_2)x(t)\|_1 \leq n\epsilon. \tag{4.40}$$

Note that Assumption 12 holds. It then follows from Lemma 43 that

$$\begin{aligned}\lim_{t\to\infty}\sup \|(\Omega\otimes I_m)x(t)\|_1 &= \lim_{t\to\infty}\sup \sum_{k=1}^{n}\left\| x_k(t) - \frac{1}{n}\sum_{j=1}^{n} x_j(t)\right\|_1 \\ &\leq \|(P\mathcal{L})^{+}\otimes I_m\|_1 n\epsilon 2^{m-1}\sum_{i=1}^{m} i!\left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2\rfloor}\right).\end{aligned} \tag{4.41}$$

In what follows, the term $(\mathbf{1}_n^T \otimes I_m)(x-r)$ is analyzed. The derivative of $(\mathbf{1}_n^T \otimes I_m)(x-r)$ can be calculated as follows

$$\begin{aligned}\frac{\mathrm{d}\left[(\mathbf{1}_n^T \otimes I_m)(x-r)\right]}{\mathrm{d}t} &= (\mathbf{1}_n^T \otimes I_m)(\dot{x}-\dot{r}) \\ &= (\mathbf{1}_n^T \otimes I_m)\{[I_n \otimes (A+BK_1)](x-r) - \beta(I_n \otimes B)\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]\} \\ &= [\mathbf{1}_n^T \otimes (A+BK_1)](x-r) - \beta(\mathbf{1}_n^T \otimes B)\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x].\end{aligned} \tag{4.42}$$

It follows from the definitions of K_1 and K_2 that $\det[\lambda I_m - (A+BK_1)]$
$= \begin{bmatrix} \lambda & -1 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \lambda & -1 \\ C_m^0 & C_m^1 & C_m^2 & \cdots & \lambda + C_m^{m-1} \end{bmatrix} = (\lambda+1)^m$, which indicates that the eigenvalues of $A+BK_1$ all have negative real parts. Here $\det(\cdot)$ denotes the determinant of a matrix. Define the variable $S = (\mathbf{1}_n^T \otimes I_m)(x-r)$, then we can rewrite (4.42) as

$$\dot{S} = [I_n \otimes (A+BK_1)]S - \beta(\mathbf{1}_n^T \otimes B)\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]. \tag{4.43}$$

Then we can use the input-to-state stability to analyze the system (4.43) by treating the term $\beta(\mathbf{1}_n^T \otimes B)\tilde{h}[(Z_n\mathcal{L} \otimes K_2)x]$ as the input and S as the state. From Lemma 34, we know that $Z_n(t) \to P$ as $t \to \infty$, leading to $\tilde{h}\{[Z_n(t)\mathcal{L} \otimes K_2]x(t)\} \to \tilde{h}[(P\mathcal{L} \otimes K_2)x(t)]$. It has been proved that $|p_i \sum_{j \in \mathcal{N}_i} a_{ij} K_2[x_i(t) - x_j(t)]| \leq \epsilon, \forall i \in \mathcal{V}$, is reached as $t \to \infty$, therefore we have $\tilde{h}\{[Z_n(t)\mathcal{L} \otimes K_2]x(t)\} \to \frac{[(P\mathcal{L} \otimes K_2)x(t)]}{\epsilon}$ as $t \to \infty$. It thus follows that $\beta(\mathbf{1}_n^T \otimes B)\tilde{h}\{[Z_n(t)\mathcal{L} \otimes K_2]x(t)\} \to \beta\frac{(\mathbf{1}_n^T P\mathcal{L} \otimes BK_2)x(t)}{\epsilon} = \mathbf{0}_m$ as $t \to \infty$, which gives $S \to \mathbf{0}_m$

and thus $\lim_{t\to\infty} \sum_{i=1}^{n}[x_i(t) - r_i(t)] \to \mathbf{0}_m$ as $t \to \infty$. It follows from (4.41) that

$$\begin{aligned}&\lim_{t\to\infty} \sup \sum_{k=1}^{n} \left\| x_k(t) - \frac{1}{n}\sum_{j=1}^{n} r_j(t) \right\|_1 \\ \leq &\|(P\mathcal{L})^{+} \otimes I_m\|_1 2^{m-1} n\epsilon \sum_{i=1}^{m} i! \left(\prod_{j=0}^{i-1} C_j^{\lfloor j/2 \rfloor} \right).\end{aligned} \tag{4.44}$$

□

Remark 45. *Both the linear and nonlinear algorithms have their unique features and advantages while with trade-offs. The advantage of the linear algorithm* (4.5) *is that it is smooth and linear and hence is easier to implement in practice. However, the trade-off is that the tracking error is zero only for reference signals whose acceleration deviations approach zero and bounded for signals with bounded acceleration deviations. On the other hand, the advantages of the nonlinear algorithm* (4.21) *are that it can achieve distributed average tracking with relatively small tracking errors for reference signals whose states and velocities are both bounded (it follows from* (4.44) *that the tracking error can be arbitrarily small by adjusting ϵ) and it can deal with more general high-order integrator systems. But the trade-off is that the nonlinear algorithm may be more "expensive" to implement than the linear one in practice.*

4.3.4 Simulations

In this section, the proposed distributed average tracking algorithms are also applied in the leader follower containment control problem, which is stated in Section 4.2.4. Similarly, we introduce an offset vector δ_i for each follower robot i and replace x_i in algorithm (4.21) with $x_i - \delta_i$. We still consider $n = 6$, and use the weight-unbalanced directed graph in Figure 4.1 as the network topology. In this case, we implement the algorithm (4.21) to illustrate Theorem 44. Here suppose the dynamics of the follower robots and the leader robots are all third-order integrators. Therefore, the state of follower robot i is $x_i = [x_i^p, \dot{x}_i^p, \ddot{x}_i^p, y_i^p, \dot{y}_i^p, \ddot{y}_i^p]^T$, where (x_i^p, y_i^p) is the X-Y coordinates of follower robot i, and the control input of follower robot i is $u_i = [u_i^x, u_i^y]^T$. Similarly, the state of leader robot i is $r_i = [r_i^x, \dot{r}_i^x, \ddot{r}_i^x, r_i^y, \dot{r}_i^y, \ddot{r}_i^y]^T$, where (r_i^x, r_i^y) is the X-Y coordinates of leader robot i, and the control input of each leader robot is denoted by $u_i^r = [u_i^{rx}, u_i^{ry}]$. As stated in Remark 40, although we only tackle the one-dimension case in Section 4.3.3, the algorithm (4.21) works in multi-dimension high-order integrators. Therefore, in this section, we consider a two-dimension third-order integrator case. We choose the following offset vectors:

$$\begin{aligned}
\delta_1 &= [2, \mathbf{0}_2^T, 3.5, \mathbf{0}_2^T]^T, & \delta_2 &= [0, \mathbf{0}_2^T, 3.5, \mathbf{0}_2^T]^T,\\
\delta_3 &= [-2, \mathbf{0}_2^T, 3.5, \mathbf{0}_2^T]^T, & \delta_4 &= [2, \mathbf{0}_2^T, -3.5, \mathbf{0}_2^T]^T,\\
\delta_5 &= [0, \mathbf{0}_2^T, -3.5, \mathbf{0}_2^T]^T, & \delta_6 &= [-2, \mathbf{0}_2^T, -3.5, \mathbf{0}_2^T]^T.
\end{aligned}$$

The dynamics of the follower robots and the leader robots are then given by $\dot{x}_i = Ax_i + Bu_i,\ \dot{r}_i = Ar_i + Bu_i^r$, where

$$A = \begin{bmatrix} A_1, \bar{\mathbf{0}}_3 \\ \bar{\mathbf{0}}_3, A_1 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix},$$

with

$$A_1 = \begin{bmatrix} 0,1,0 \\ 0,0,1 \\ 0,0,0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0,0 \\ 0,0 \\ 1,0 \end{bmatrix}, \ B_2 = \begin{bmatrix} 0,0 \\ 0,0 \\ 0,1 \end{bmatrix}.$$

Therefore, we have $K_1 = \begin{bmatrix} -1,-3,-3,0,0,0 \\ 0,0,0,-1,-3,-3 \end{bmatrix}$ and $K_2 = \begin{bmatrix} 1,2,1,0,0,0 \\ 0,0,0,1,2,1 \end{bmatrix}$. The initial states of the follower robots are chosen as $x_1(0) = [0,1,0,25,0,0]^T$, $x_2(0) = [2,1,0,20,0,0]^T$, $x_3(0) = [3,1,0,10,0,0]^T$, $x_4(0) = [2,1,0,-10,0,0]^T$, $x_5(0) = [1,1,0,-15,0,0]^T$, $x_6(0) = [0,1,0,-25,0,0]^T$, the initial states of the leader robots are chosen as $r_1(0) = [0,1,0,5,0,0]^T$, $r_2(0) = [2,1,0,5,0,0]^T$, $r_3(0) = [3,1,0,2.5,0,0]^T$, $r_4(0) = [2,1,0,-2.5,0,0]^T$, $r_5(0) = [1,1,0,-5,0,0]^T$, $r_6(0) = [0,1,0,-5,0,0]^T$. We assume the leader agent i is operating in the environment with trajectories satisfying $u_i^{rx} = 0$ and $u_i^{ry} = \cos(t) + 0.1 \times i \times \cos(t)$. The control gain β is chosen as 80.

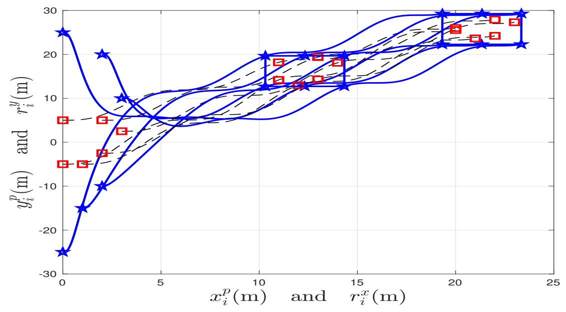

Figure 4.5: State trajectories of all the robots for the case in Section 4.3.4

The simulation results are shown in Figures 4.5-4.7. In particular, Figure 4.5 shows the X-Y coordinates of all the robots, where the blue solid lines denote the positions of the follower robots and the black dashed lines denote the positions of the leader robots. Two snapshots at 10 s and 20s, denoted by the blue stars (follower robots) and red squares (leader robots) in Figure 4.5, show that all followers follow the group of all the leader robots in a containment fashion in both cases. Figure 4.6 (respectively, Figure 4.7) shows the state trajectories of all the follower robots without the offsets and the geometric center of all the leader robots in X-coordinate (respectively, in Y-coordinate). We can see that the follower robots converge to the vicinity of the geometric center of all leader robots, which are consistent with Theorems 44.

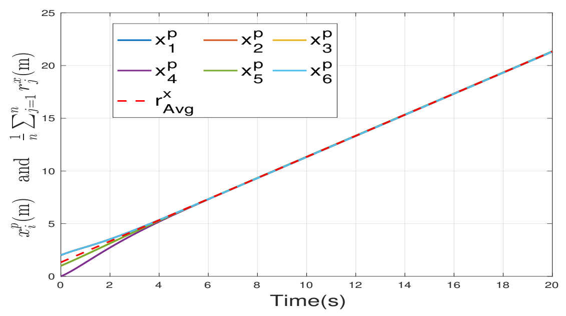

Figure 4.6: State trajectories of all the follower robots without the offsets and the geometric center of all the leader robots in X-coordinate for the case in Section 4.3.4

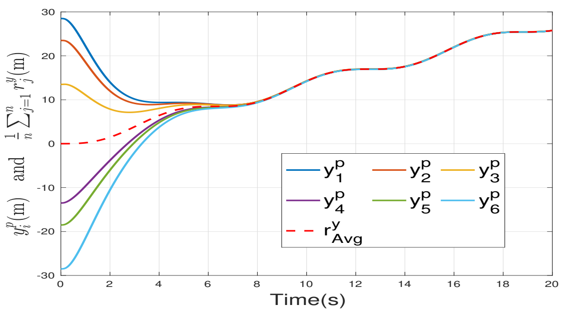

Figure 4.7: State trajectories of all the follower robots without the offsets and the geometric center of all the leader robots in Y-coordinate for the case in Section 4.3.4

4.4 Conclusions

In this chapter, we have studied distributed average tracking in weight-unbalanced directed graphs, which attempts to push a set of networked agents to track the average of the locally available time-varying reference signals, where each agent can only receive information from its neighbors. We first propose a linear algorithm for single-integrator dynamics. We have shown that the tracking error is upper bounded if the reference signals have bounded acceleration deviations. We also investigate a nonlinear algorithm for high-order integrator dynamics, which guarantees that distributed average tracking can be achieved with arbitrarily small tracking errors if the reference signals and their velocities are all bounded, and the control gain is properly chosen.

Chapter 5

Distributed Packet Routing Using Reinforcement Learning

In this chapter, we consider a path optimization problem, specifically for packet routing, in large complex networks. We develop and evaluate a model-free approach, applying multi-agent meta reinforcement learning (MAMRL) that can determine the next-hop of each packet to get it delivered to its destination with minimum time overall. Specifically, we propose to leverage and compare deep policy optimization RL algorithms for enabling distributed model-free control in communication networks and present a novel meta-learning-based framework, MAMRL, for enabling quick adaptation to topology changes. To evaluate the proposed framework, we simulate with various WAN topologies. Our extensive packet-level simulation results show that compared to classical shortest path and traditional reinforcement learning approaches, MAMRL significantly reduces the average packet delivery time even when network demand increases; and compared to a non-meta deep policy optimization algorithm, our results show the reduction of packet loss in much fewer episodes when link failures occur while offering comparable average packet delivery time.

5.1 Problem Formulation

In the packet routing problem, packets are transmitted from a source to its destination through intermediate routers and available links. The mathematical model is given below.

Environment. We consider a possibly time-varying communication network environment, which is characterized by an undirected graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$, where $\mathcal{V} = \{1, \cdots, n\}$ is a set of routers and $\mathcal{E}_t \subseteq \mathcal{V} \times \mathcal{V}$ are transmission links between the routers at time t. The bandwidth of each link is limited and packet loss might occur when the size of the packet to be transmitted is greater than the link's capacity. The communication network is possibly time varying since link failures might happen during working hours. When the link failure happens, the capacity of the link becomes zero. Each router i has a set of neighbor routers denoted by $\mathcal{N}_i(t) = \{j \in \mathcal{V} : (i, j) \in \mathcal{E}_t\}$.

Routing. Packets are introduced into the network with a node of origin and another node of destination. They travel to their destination nodes by hopping on intermediate nodes. Each router only has one local port/queue used to store traffic. The queue of routers follows the first-in-first-out (FIFO) criterion. The node can forward the top packet in its local queue to one of its neighbors. Once a packet reaches its destination, it is removed from the network.

Objective. The packet routing problem aims at finding the optimal transmission path between source and destination routers to minimize the average packet delivery time, which is the sum of queuing time and transmission time while preventing packet loss when link failures happen.

5.2 Background

5.2.1 Performance under Partial Observability

Figure 2.1 shows the general process of reinforcement learning, where the agent is able to observe the global information of the environment. In this work, we consider a path optimization problem in the distributed network environment, indicating that each router only has access to its own information and the information received from its adjacent routers. It follows that the path optimization problem can be modeled as a multi-agent partially observable Markov decision process (POMDP). A POMDP, referred as $\mathcal{M}$, for n routers is defined by a tuple $\langle \mathcal{S}, \{\mathcal{O}^i\}_{i\in\mathcal{V}}, \{\mathcal{A}^i\}_{i\in\mathcal{V}}, P, \{R^i\}_{i\in\mathcal{V}}\rangle$, where S and P carry the same meaning as those in Section 2.5 and $\mathcal{V}$ denotes the set of all routers. $\mathcal{O}^i$, $\mathcal{A}^i$, and R^i are the local observation space, local action space and local reward function of router i, respectively. Then we have $\mathcal{A} = \Pi_{i=1}^{n}\mathcal{A}^i$ is the joint action space of all routers. Each router only has access to a private local observation correlated with the state o_t^i. To choose actions, each router i uses a stochastic parametric policy $\pi_{\theta^i}^i : \mathcal{O}^i \times \mathcal{A}^i \to [0,1]$, where $\pi_{\theta^i}^i(a_t^i|o_t^i)$ represents the probability of choosing action a_t^i at observation o_t^i. Thus, the joint policy of all routers $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0,1]$ satisfies $\pi_\theta(a_t|s_t) = \Pi_{i=1}^{n}\pi_{\theta^i}^i(a_t^i|o_t^i)$. For a given time horizon H we define the trajectory $\tau := (s_0, a_0, \cdots, s_H, a_H, s_{H+1})$ as the collection of state action pairs ended at time $t = H$. The probability distribution of the initial state is denoted by $\rho(s_0)$. In the path optimization problem (cooperative multi-agent problem), the collective objective of all the routers is to collaboratively find policies $\pi^i_{\theta^i}$ for all $i \in \mathcal{V}$ that maximize the globally expected trajectory reward over the whole network. The goal of all routers is as follows,

$$\max_{\theta^i, i \in \mathcal{V}} J(\theta) = \mathbb{E}_\tau \left[R(\tau) \right], \tag{5.1}$$

where

$$R(\tau) = \sum_{t=0}^{H} r^i_t = \sum_{t=0}^{H} \left(\tilde{r}^i_t + \frac{1}{n} \sum_{i=1}^{n} \hat{r}^i_t \right),$$

and r^i_t denotes the reward needed by router i at time t. r^i_t consists of two parts: 1) $\tilde{r}^i_t$ denotes the reward signal based solely on individual behavior, and 2) $\frac{1}{n}\sum_{i=1}^{n} \hat{r}^i_t$ denotes the reward signal based on global behavior. Note that only $\tilde{r}^i_t$ and $\hat{r}^i_t$ can be known by router i in a partially observable environment.

As stated in Section 2.5, in this work, we investigate how deep policy optimization algorithms work in path optimization problems. The main idea is to directly adjust the parameters $\theta_i, i \in \mathcal{V}$ of the policies in order to maximize the objective in (5.1) by taking steps in the direction of $\nabla_{\theta_i} J(\theta_i, \cdots, \theta_n)$. For POMDP, the gradient of the expected return for router i can be written[1] as,

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_\tau \left[\sum_{t=0}^{H} \nabla_{\theta^i} \log \pi^i_{\theta^i}(a^i_t | o^i_t) R(\tau) \right]. \tag{5.2}$$

Note that with only local information, function $R(\tau)$ cannot be well estimated since the estimation requires the reward $\hat{r}^i$ of all routers. In this work, we propose to use a dynamic consensus algorithm to estimate $R(\tau)$ using only local information, described in Section 5.3.

[1]The derivation of Equation (5.2) is provided in Appendix .1.

5.2.2 Model-agnostic Meta-learning

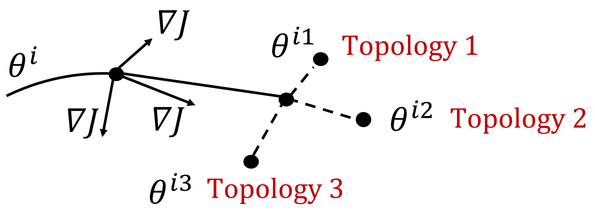

Figure 5.1: The framework of model-agnostic meta learning.

In this work, we consider path optimization in the presence of link failures. It follows that, once there is a link failure, the state transition function of the environment changes accordingly, indicating that a new POMDP occurs. Let $\mathcal{M}_0$ denote the Markov process modeled by the full network environment (no link failures) and $\mathcal{M}_k$, where $k > 0$, denote the Markov process modeled by the network environment with different link failure scenarios. Suppose that the distribution of all POMDPs follows $\eta(\mathcal{M})$. To make the routing algorithm adapt to link failures (different POMDPs) quickly, we leverage the model-agnostic meta-learning [30] into the policy optimization algorithms. Meta-reinforcement aims to learn an algorithm that can quickly learn optimal policies in $\mathcal{M}_k$ drawn from a distribution $\eta(\mathcal{M})$ over a set of Markov decision processes. Our approach trains a well-generalized parametric policy initialization that is close to all the possible environments (POMDPs), such that it can quickly improve its performance on a new environment with one or a few vanilla policy gradient steps (see Figure 5.1). The meta-learning objective can be written as:

$$
\begin{aligned}
\max_{\theta} \quad & E_{\mathcal{M}\sim\eta(\mathcal{M}),\tau\sim p(\tau|\pi_{\hat{\theta}})}\left[R(\tau)\right] \\
\text{s.t.} \quad & \hat{\theta} = \theta + \alpha\nabla_{\theta}\mathbb{E}_{\tau\sim p(\tau|\pi_{\theta})}[R(\tau)],
\end{aligned}
\tag{5.3}
$$

where $\alpha > 0$ is the learning rate and $p(\tau|\pi_\theta)$ represents the distribution of trajectory τ given policy π_θ. Model-agnostic meta-learning attempts to learn an initialization θ^* such that for any environment $\mathcal{M}_k$ the policy attains maximum performance after a few policy gradient steps.

5.3 Design: MAMRL Approach

Our standard RL setup consists of multiple router agents interacting with an environment (communication networks) in discrete decision epochs. We investigate the deep policy optimization algorithm to address packet routing in a partially observable network environment. To make the router controllers adapt to link failures more quickly, we leverage the model-agnostic meta-learning technique to learn the well-generalized policy initialization.

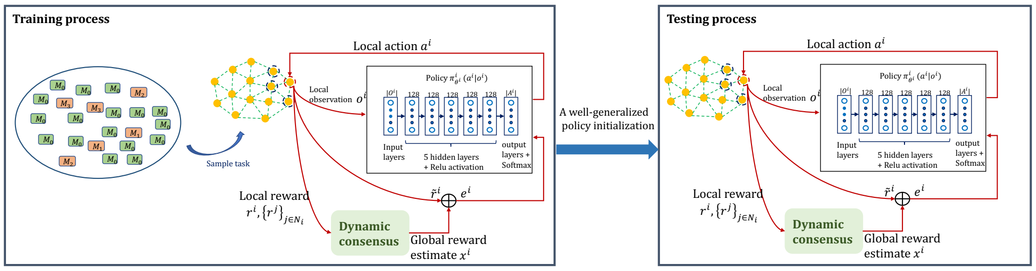

Figure 5.2: MAMRL framework.

Figure 5.2 shows the MAMRL setup (training and testing process) per router. In the testing process, each router uses the deep policy optimization algorithm coupled with the dynamic consensus estimator to learn the optimal policy. In order to let the routers adapt to topology changes quickly, the policy of each router is initialized using the well-generalized policy initialization, which is the output of the training process. The training process follows the traditional model-agnostic meta-reinforcement learning framework. The basic idea is letting the network controller encounter multiple link failures in the training process. It can use this experience to learn how to adapt if similar situations occur while deployed. In Figure 5.2, $\mathcal{M}_0$ denotes the Markov process modeled by the full network environment (no link failures) and $\mathcal{M}_k$, where $k > 0$, denotes the Markov process modeled by the network environment with link failures. In the training process, the network controller collects data samples from all possible network environments according to the distribution $\eta(\mathcal{M})$. However, the traditional design of model-agnostic meta-learning mainly focuses on single-agent (centralized) reinforcement learning problems. How to solve a multi-agent RL problem using model-agnostic meta-learning in a distributed manner is rarely studied. In this work, we aim to train and execute the network controller in a distributed manner. As shown in Figure 5.2, each router has an independent policy model that is represented by a deep neural network. The core of the proposed control framework is letting each router run a deep reinforcement learning algorithm to find the best action at each decision time instant, using only local information and local interaction. Since the routers aim to minimize the average packet delivery time of the whole network, each router needs to feed the global packet delivery time into its policy model as feedback/reward. To achieve this goal, we leverage the dynamic consensus algorithm to estimate the global reward function.

The policy optimization algorithms aim to find the best policy parameters that produce the highest long-term expected return using gradient ascent. The gradient of the long-term expected return for the parameters of each router's policy is defined in Equation (5.2). However, with only local information, function $R(\tau)$ cannot be well estimated since the estimation requires the rewards $\hat{r}^i$ of all routers $\forall i \in \mathcal{V}$. This motivates our consensus-based policy gradient algorithm that leverages the communication network to diffuse the local information, fostering collaboration among routers. We adapt the following dynamic consensus algorithm [50] into the policy optimization method.

$$\begin{aligned} x_t^i &= \hat{r}_t^i - y_t^i, \\ y_{t+1}^i &= \beta \sum_{j \in \mathcal{N}_i} (x_t^i - x_t^j) + y_t^i, \end{aligned} \tag{5.4}$$

where $0 < \beta < 1$ is the control gain, x_t^i and y_t^i are local estimators, and $\mathcal{N}_i$ denotes the neighbor sets of router i. It can be proved that x_t^i converges to the vicinity of $\frac{1}{n}\sum_{i=1}^n \hat{r}_t^i$ within a few time steps. It is worthy to mention that only local information is used in the designed estimator Equation (5.4).

We develop the following policy optimization method for POMDP,

$$\nabla_{\theta^i} \bar{J}(\theta^i) = \mathbb{E}_\tau \left[\sum_{t=0}^{H} \nabla_{\theta^i} \log \pi_{\theta^i}^i (a_t^i | o_t^i) \bar{R}^i(\tau) \right], \tag{5.5}$$

where

$$\bar{R}^i(\tau) = \sum_{l=0}^{H} e_l^i = \sum_{l=0}^{H} \tilde{r}_l^i + x_l^i \approx \sum_{l=0}^{H} \left(\tilde{r}_l^i + \frac{1}{n} \sum_{i=1}^{n} \hat{r}_l^i \right). \tag{5.6}$$

Here, e_l^i denotes the sum of the local reward signal $\tilde{r}_l^i$ and global reward estimate x_l^i. And x_l^i is obtained by the dynamic consensus estimator designed in Equation (5.4). Note that both $\tilde{r}^i$ and x^i can be obtained locally.

We build the deep neural network with one input layer, five hidden layers of size 128 with ReLU, and one output layer with Softmax (see Figure 5.2). As shown in Figure 5.2, at each decision epoch t, each router i provides the local observation o_t^i to the policy model π_{θ^i} and gets the action a^i back. Router i performs action a_t^i and switch to a new state. Then router i feeds the local reward $\tilde{r}_{t+1}^i$ and global reward estimate x_{t+1}^i, which is the output of the dynamic consensus estimator, to the policy model and the policy model π_{θ^i} updates its weights θ^i with respect to the received reward estimate e^i. It is worthwhile to mention that to update the policy in the direction of greater cumulative reward using Equation (5.5), only local information o_t^i, a_t^i and e_t^i are required. By integrating model-agnostic meta-learning and the proposed multi-agent policy optimization algorithm, MAMRL for packet routing problem, where both training and execution process is distributed. These are shown in Algorithms 1 and 2.

We design the local observation o_t^i, local action a_t^i and local estimation of the reward function e_t^i below,

- **Observation of router i,** o_i: 1) destination router of first packet in the local queue; 2) the last ten step actions taken by router i; 3) the address of the router which has the longest queue among all the neighbor router of router i.

- **Action of router i,** a_i: next hop of current packet in the queue.

- **Reward estimate of router i,** e^i: sum of $\tilde{r}^i$ and x^i, where $\tilde{r}^i$ is negative number of packet loss occurred at router i and x^i is the estimate of $\frac{1}{n}\sum_{j=1}^{n}\hat{r}^j$ using Equation (5.4). Here, $\hat{r}^j$ denotes the negative average delivery time of all the packets delivered

Algorithm 1: Multi-agent meta reinforcement learning algorithm (MAMRL train time)

```
Input: $\eta(\mathcal{M})$: distributions of network environments;
Input: $\alpha$: step size hyper-parameter;
randomly initialize $\theta^i, i \in \mathcal{V}$;
while not done do
    sample batch of environments $\mathcal{M}_k \sim \eta(\mathcal{M})$;
    for all $\mathcal{M}_k$ do
        for all routers $i \in \mathcal{V}$ do
            Sample $K$ trajectories $D^i = \{(o_0^i, a_0^i, e_0^i, \cdots, o_H^i, a_H^i, e_H^i)\}$ using $\pi_{\theta^i}^i$
            and Equation (5.4) in $\mathcal{M}_k$;
        end
        for all routers $i \in \mathcal{V}$ do
            Evaluate $\nabla_{\theta^i} \bar{J}(\theta)$ using $D^i$ based on Equation (5.5);
            Compute adapted parameters with gradient descent:
            $\hat{\theta}^i = \theta^i - \alpha \nabla_{\theta^i} \bar{J}(\theta)$;
        end
        for all routes $i \in \mathcal{V}$ do
            Sample $K$ trajectories $\hat{D}^i = \{(o_0^i, a_0^i, e_0^i, \cdots, o_H^i, a_H^i, e_H^i)\}$ using $\pi_{\hat{\theta}^i}^i$
            and Equation (5.4) in $\mathcal{M}_k$;
        end
    end
    for all routes $i \in \mathcal{V}$ do
        Evaluate $\nabla_{\hat{\theta}^i} \bar{J}(\hat{\theta})$ using $\{\hat{D}^i\}_{i \in \mathcal{V}}$ based on Equation (5.5);
        Update $\theta^i$ with gradient descent: $\theta^i = \theta^i - \alpha \nabla_{\hat{\theta}^i} \bar{J}(\hat{\theta})$;
        using $\hat{D}^i$ based on Equation (5.5);
    end
end
Return $\theta^i$, $i \in \mathcal{V}$ as parameter initialization.
```

Algorithm 2: Multi-agent meta reinforcement learning algorithm (MAMRL test time)

```
Input: A dynamic network environment with possible task distribution $\eta(\mathcal{M})$;
Input: $\alpha$: step size hyper-parameter;
Input: Learned parameter initialization $\theta^i, i \in \mathcal{V}$;
while not done do
    if Link failure is True then
        $\theta_t^i \leftarrow \theta^i$, $i \in \mathcal{V}$
    end
    for all routers $i \in \mathcal{V}$ do
        Sample $K$ trajectories $D = \{(o_0^i, a_0^i, e_0^i, \cdots, o_H^i, a_H^i, e_H^i)\}$ using $\pi_{\theta^i}^i$ and
        Equation (5.4);
    end
    Evaluate $\nabla_{\theta^i} \bar{J}(\theta^i)$ using $D$ based on Equation (5.5);
    for all routers $i \in \mathcal{V}$ do
        Compute adapted parameters with gradient descent:
        $\theta_{t+1}^i = \theta_t^i - \alpha \nabla_{\theta^i} \bar{J}(\theta^i)$;
    end
end
```

to router j.

Note that the design of state space and reward is critical to the success of a deep reinforcement learning method. Our design of the state space captures key components of the network environment. For our design of the reward function, element $\frac{1}{n}\sum_{j=1}^{n}\hat{r}^j$ is introduced to minimize the average packet delivery time of the whole network, element $\tilde{r}^i$ is included to minimize the packet loss occurred at router i in the presence of link failures. Note that in our design $e^i = x^i + \tilde{r}^i$, where $\tilde{r}^i$ is the negative number of packet loss occurred only at router i but x^i is the estimate of the negative average delivery time of the whole network environment. The reasons are summarized below.

Optimizing for packet delivery time: To achieve this goal, all the routers need to collaboratively find the best paths to reroute the traffic. And the delivery time of the packets that are delivered to router i is determined by the decisions of all the intermediate

Topology Name	Number of nodes	Number of edges
B4	12	19
Geant	21	32
ATT	25	56

Table 5.1: Network topologies used in our evaluations.

routers. That is, packet delivery time is a signal based on global behavior, it is not enough for router i to only know the delivery time of the packets delivered to itself.

Optimizing for link failures: Although this goal also involves reading packet loss in the whole network, the link failures have little effect on the routers that are not directly connected to the failed links. Therefore, in our design, we only provide the packet loss that occurred at router i to the policy π_{θ^i} as the feedback.

5.4 Evaluation

We conduct extensive simulations to evaluate the performance of the proposed MAMRL framework in a path optimization problem with static topologies and topologies with possibly failed links.

We evaluate the results to,

- Benchmark the RL techniques against existing path optimization approaches, and
- Show how quickly MAMRL adapts to link failures.

The simulation runs are performed on three network topologies, B4, Geant, and ATT network. See Table 5.1 for a specification of network sizes. The B4 and ATT topologies (link capacities) and their traffic matrices (packet size) were obtained from the authors of

Teavar [10]. The Geant topology is the European Research network providing connectivity to science experiments across Europe and US labs (www.geant.org).

To model the packet arrival, a discrete event network simulator is developed, based on Open AI gym. Packets are introduced into the network with a node of origin and another node of destination. The packet arrives according to the Poisson process of rate λ. They travel to their destination node by hopping on intermediate nodes. Each router only has a one local port/queue used to store traffic. The queue of routers follows the FIFO criterion. In each time unit, the node forwards the top packet in its local queue to one of its neighbors. Once a packet reaches its destination, it is removed from the network environment. The bandwidth of each link is limited and packet loss might occur when the size of the packet to be transmitted is greater than the link's capacity. When the link failure happens, the capacity of the link becomes zero.

In the experiments, we choose the step size α as 0.01. In addition, we use trust-region policy optimization (TRPO) [85] as the meta-optimizer and the standard linear feature baseline [24] is used.

•Impact of Increasing Network Load

We first test the MAMRL algorithm with static topologies (no link failures). We compare with the classical shortest path algorithm and two existing RL-based routing algorithms:

- Shorted path algorithm (SPA) [103]: a traditional packet routing algorithm.
- Q-routing [12]: a value-based reinforcement learning algorithm.
- Policy gradient (PG) [77]: a policy-based reinforcement learning algorithm.

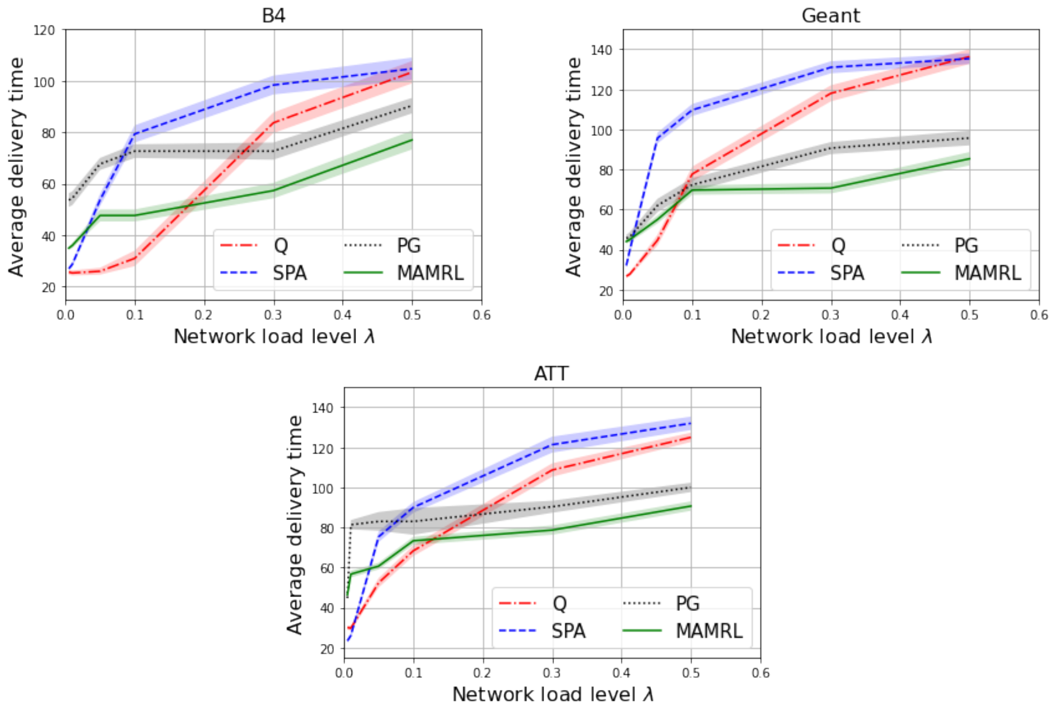

Figure 5.3: Comparing average packet delivery time as load increases.

In the experiments, the episodes terminate at the horizon of $H = 500$. After 10000 training episodes, we restored the well-trained models to compare their performance in a new test environment where packets were generated at the corresponding network load level. Note that the SPA does not need training and can be applied to test directly. We tested the network on loads ranging from 0.005 to 0.5 and measured the average packet delivery time of several episodes in the testing process to compare with the results given by the above-mentioned baseline controllers. The load corresponds to the value of λ of the Poisson arrival process for the average number of packets injected per unit time.

The average packet delivery time results versus different network load are shown in Figure 5.3. Under conditions of low load for all the three topologies, MAMRL is slightly inferior to Q-routing and SPA. As the load increases, the MAMRL performs much better than the baseline algorithms. On the B4 topology, when the traffic load is high (i.e., $\lambda = 0.5$), MAMRL reduces the average packet delivery time by 25%, 24%, and 14%, respectively, compared to SPA, Q-routing, and policy gradient algorithms. On the Geant topology, when the traffic load $\lambda = 0.5$, MAMRL significantly reduces the average packet delivery time by 37%, 37%, and 10%, respectively, compared to SPA, Q-routing, and policy gradient algorithms. And on the ATT topology, when the traffic load $\lambda = 0.5$, MAMRL reduces the average packet delivery time by 33%, 28%, and 10%, respectively, compared to SPA, Q-routing, and policy gradient algorithms. The reason is described as follows. Under conditions of low load, there is no congestion along the route. Therefore, the deterministic policy learned by Q-routing performs as well as SPA, which is the optimal routing policy under low load. However, the routing policy learned by MAMRL is stochastic, which means that not all of the packets are sent down the optimal link. That is why the performance of MAMRL is slightly inferior to Q-routing and SPA under low load. As the load increases, the routes are getting crowded and the length of the queues is getting longer. Due to the stochastic nature of the communication network environment, the optimal policy should be stochastic under conditions of high load. This explains why MAMRL performs much better than the Q-routing and SPA controllers under high load. The results in [77] also show that policy-based reinforcement learning algorithm performs better than value-based algorithms, especially on high flow load. However, the work in [77] only considers a simple policy gradient algorithm for the packet routing problem. Instead, we investigate a deep policy optimization algorithm that can take much more information as its inputs, enlarging the state-action space for better policy making. The results in Figure 5.3 indicate that our MAMRL algorithm achieves a shorter delivery time than a simple policy gradient algorithm.

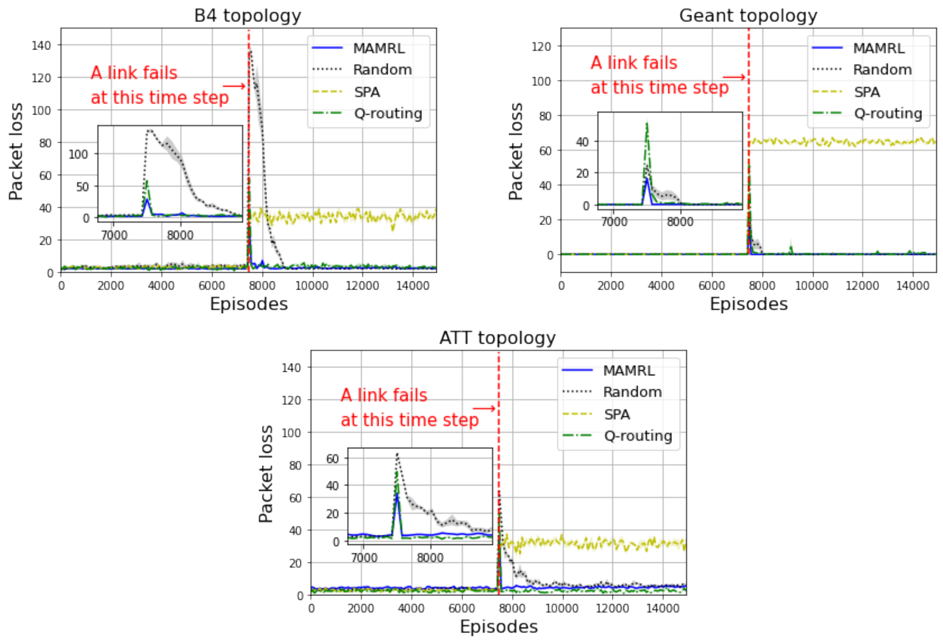

Figure 5.4: Packet loss results in the presence of link failures.

•Impact of Link Failures

Next, we test the MAMRL algorithm in the presence of link failures with network load $\lambda = 0.3$. We let the router train and encounter all possible network environments (link failure scenarios) according to the distribution $\eta(\mathcal{M})$ and return policy parameters using Algorithm 1. We restored the well-trained models in a new test environment where the links get disconnected according to the distribution $\eta(\mathcal{M})$ (We assume that only one link gets failed at one time). We compare results of using the following three controllers: (1) testing the policy from the initialization parameters obtained by MAMRL, (2) testing the policy from randomly initialized weights (called random in the following), (3) shortest path algorithm (SPA) [103], and (4) Q-routing algorithm [12]. Figure 5.4 show the results of the packet loss versus episodes and Figure 5.5 show the results of the average packet delivery time versus episodes. Also, we show the performance of reinforcement learning routing algorithms (MAMRL, random, Q-routing) over the three network topologies during the online learning procedure in terms of the reward. We present the corresponding simulation results in Figure 5.6. We can make the following observations from these results.

1. In Figure 5.4, when there is a link failure, the model-based routing algorithm (i.e., SPA) witnesses a huge packet loss. The reason is that the SPA algorithm relies on previous knowledge of the network topology to make decisions. Here we assume that as the networks grows, it becomes longer to update the ISIS/OSPF protocols for link failures and update the tables. Both ISIS/OSPF use the same Dijkstra algorithm for computing the best path through the network. The other learning algorithms (MAMRL, random, Q-routing) are model-free controllers and the policy of the model-free controller. When link failure happens, the packet loss sensor will tell the RL routing controllers that there are many packet loss at the particular link. Based on our design, the packet loss hurts the reward of the RL routing controllers. To maximize the reward function, the RL routing controller will adjust their policies to improve the reward function and hence reduce the packet loss accordingly.

2. Figure 5.6 shows how the reward value changes during online learning over the three network topologies. It is seen that when there is a link failure, for the B4 topology, Q-routing adapts to the link failure (reward values converge to the stable states) after about 30 episodes, MAMRL (our algorithm) adapts to the link failure after about 35

Topology Name	Q-routing	MAMRL	Random
B4	23	25	800
Geant	35	35	100
ATT	29	30	1000

Table 5.2: Average number of episodes used to adapt to link failures.

episodes and random algorithm (deep policy optimization with randomly initialized weights) adapts to the link failure after about 800 episodes. The results for Geant topology and ATT topology are shown in Table 5.2. Q-routing is based on a value-based Q-learning algorithm and is often much faster to learn a policy than policy optimization algorithms [72]. In this work, we propose the MAMRL algorithm which leverages model-agnostic meta-learning to help the policy optimization adapt to link failures quickly. The basic idea of MAMRL is letting the network controller encounter all possible link failures in the training process. It can then use that experience to learn how to adapt. MAMRL aims to learn a well-generalized policy initialization that is close to all possible situations of the environment. Whenever there are continual packet losses at a particular link, the MAMRL controller will reinitialize the policy models based on the pre-trained well-generalized policy initialization. It can be seen from Figure 5.6, the MAMRL controller adapts to link failures with a speed that is comparable to the Q-routing algorithm. However, the normal policy optimization controller adapts to the link failures much more slowly.

Policy optimization algorithms use gradient descent to optimize an optimization problem. And traffic engineering aims at finding a solution to forward the data traffic to maximize a utility function. The utility function might concern a set of values. In our design, the objective is to minimize the packet delivery time and packet loss, therefore, the utility function, which corresponds to the reward function in the RL algorithms, consists of a function of packet loss and a function of packet delivery.

In future works, we can add multiple objectives such as bandwidth utilization, latency and more, if we want the RL controller to optimize on a number of multiple parameters.

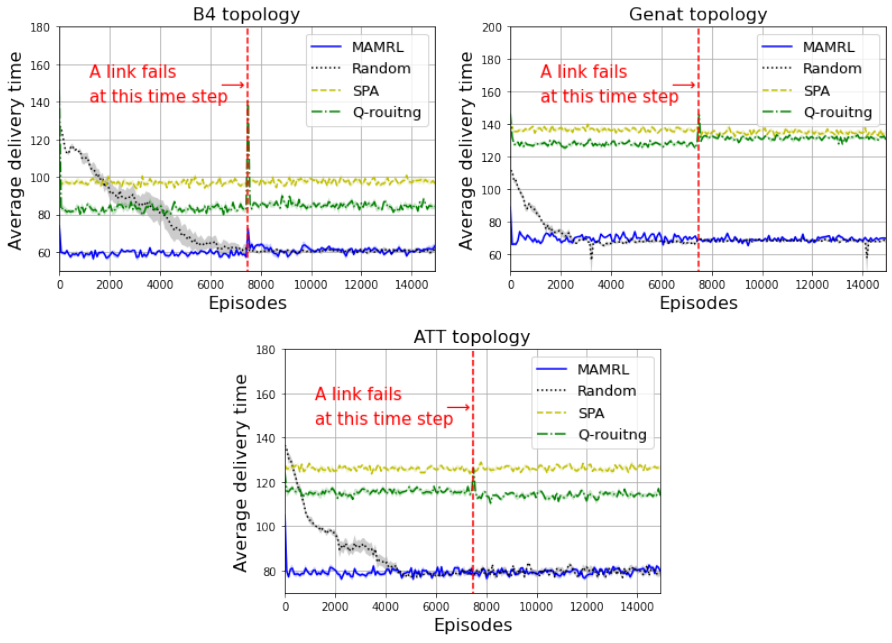

Figure 5.5: Average packet delivery time results in the presence of link failures.

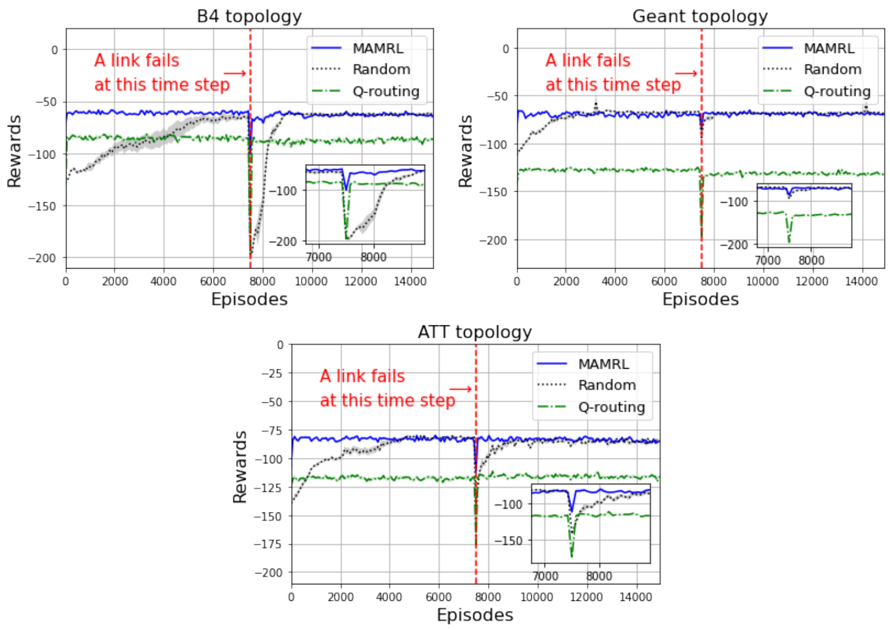

Figure 5.6: Reward in the presence of link failures.

Chapter 6

Summary

In this dissertation, we have studied two distributed optimization problems for multi-agent systems: distributed continuous-time optimization with time-varying objective functions, 2) packet routing for communication networks.

We have proposed multiple algorithms for different application scenarios, more specifically,

- We address distributed continuous-time optimization problems with convex time-varying objective functions for undirected graphs. First, for the unconstrained case, a distributed nonsmooth algorithm coupled with a sate-dependent gain is proposed. This algorithm can solve the time-varying optimization problem without imposing a bound on any information about the local objective functions. Then, we investigate distributed constrained time-varying optimization problems and propose three distributed algorithms, respectively, for 1) the case where there only exist common time-varying linear equality constraints, 2) the case where there exist only time-varying nonlinear inequality constraints, and the case where there exist not only time-varying nonlinear inequality constraints but also linear equality constraints.

- We study distributed time-varying optimization with quadratic objective function, which is equivalent to distributed average tracking problem. We seek a design methodology for distributed average tracking under possibly unbalanced graphs and propose two distributed algorithms, respectively, for single-integrator and high-order integrator dynamics.

- We consider packet routing problem for distributed communication networks. We aim to find optimal paths for the packets in the presence of link failures and propose to leverage policy optimization reinforcement learning algorithms for enabling quick adaption to topology changes.

Bibliography

[1] https://www.vicon.com/.

[2] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad hoc networks*, 2(1):1–22, 2004.

[3] Hasan AA Al-Rawi, Ming Ann Ng, and Kok-Lim Alvin Yau. Application of reinforcement learning to routing in distributed wireless networks: a review. *Artificial Intelligence Review*, 43(3):381–416, 2015.

[4] Ramy E Ali, Bilgehan Erman, Ejder Baştuğ, and Bruce Cilli. Hierarchical deep double q-routing. In *Proceedings of the International Conference on Communications*, pages 1–7, online, 2020.

[5] Tom M Apostol and CM Ablow. Mathematical analysis. *Physics Today*, 11(7):32, 1958.

[6] Mohammad Mehdi Asadi, Stephane Blouin, and Amir G Aghdam. Distributed dynamic average consensus in asymmetric networks. In *Proceedings of the American Control Conference*, pages 864–869, Milwaukee, USA, 2018.

[7] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. *RFC3272*, 2002.

[8] He Bai, Randy A Freeman, and Kevin M Lynch. Robust dynamic average consensus of time-varying inputs. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3104–3109, Atlanta, USA, 2010.

[9] Dennis S Bernstein. *Matrix mathematics: theory, facts, and formulas*. Princeton: Princeton university press, 2009.

[10] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. Teavar: striking the right utilization-availability balance in wan traffic engineering. In *Proceedings of the ACM SIGCOMM*, pages 29–43. Beijing, China, 2019.

[11] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):16, 2018.

[12] Justin A Boyan and Michael L Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in neural information processing systems*, pages 671–678, Denver, USA, 1993.

[13] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge: Cambridge university press, 2004.

[14] Armir Bujari, Claudio E Palazzi, and Daniele Ronzani. A comparison of stateless position-based packet routing algorithms for fanets. *IEEE Transactions on Mobile Computing*, 17(11):2468–2482, 2018.

[15] Fei Chen, Yongcan Cao, and Wei Ren. Distributed average tracking of multiple time-varying reference signals with bounded derivatives. *IEEE Transactions on Automatic Control*, 57(12):3169–3174, 2012.

[16] Fei Chen, Gang Feng, Lu Liu, and Wei Ren. Distributed average tracking of networked Euler-Lagrange systems. *IEEE Transactions on Automatic Control*, 60(2):547–552, 2015.

[17] Fei Chen and Wei Ren. A connection between dynamic region-following formation control and distributed average tracking. *IEEE Transactions on Cybernetics*, 48(6):1760–1772, 2018.

[18] Fei Chen, Wei Ren, Weiyao Lan, and Guanrong Chen. Distributed average tracking for reference signals with bounded accelerations. *IEEE Transactions on Automatic Control*, 60(3):863–869, 2015.

[19] Chunhsiang Cheng, Ralph Riley, Srikanta PR Kumar, and Jose J Garcia-Luna-Aceves. A loop-free extended bellman-ford routing protocol without bouncing effect. *ACM SIGCOMM Computer Communication Review*, 19(4):224–236, 1989.

[20] Ashish Cherukuri and Jorge Cortes. Initialization-free distributed coordination for economic dispatch under varying loads and generator commitment. *Automatica*, 74:183–193, 2016.

[21] Samuel PM Choi and Dit-Yan Yeung. Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control. In *Advances in Neural Information Processing Systems*, pages 945–951, Denver, USA, 1996.

[22] Jorge Cortes. Discontinuous dynamical systems. *IEEE Control Systems Magazine*, 28(3):36–73, 2008.

[23] Jorge Cortés. Distributed Kriged Kalman filter for spatial estimation. *IEEE Transactions on Automatic Control*, 54(12):2816–2827, 2009.

[24] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the International Conference on Machine Learning*, pages 1329–1338, New York City, USA, 2016.

[25] Amit Dvir and Athanasios V Vasilakos. Backpressure-based routing protocol for dtns. In *Proceedings of the ACM SIGCOMM*, pages 405–406, New Delhi, India, 2010.

[26] Christopher Edwards and Sarah Spurgeon. *Sliding mode control: theory and applications.* London: Crc Press, 1998.

[27] Mahyar Fazlyab, Santiago Paternain, Victor M Preciado, and Alejandro Ribeiro. Prediction-correction interior-point method for time-varying convex optimization. *IEEE Transactions on Automatic Control*, 63(7):1973–1986, 2017.

[28] Zhi Feng and Guoqiang Hu. Finite-time distributed optimization with quadratic objective functions under uncertain information. In *Proceedings of the IEEE Conference on Decision and Control*, pages 208–213, Melbourne, Australia, 2017.

[29] Aleksei Fedorovich Filippov. *Differential equations with discontinuous righthand sides: control systems.* Berlin: Springer Science & Business Media, 2013.

[30] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the International Conference on Machine Learning*, pages 1126–1135, 2017.

[31] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 1146–1155, Sydney, Australia, 2017.

[32] Randy A Freeman, Peng Yang, and Kevin M Lynch. Stability and convergence properties of dynamic average consensus estimators. In *Proceedings of the IEEE Conference on Decision and Control*, pages 338–343, San Diego, USA, 2006.

[33] Jemin George and Randy A Freeman. Robust dynamic average consensus algorithms. *IEEE Transactions on Automatic Control*, 64(11):4615–4622, 2019.

[34] Sheida Ghapani, Salar Rahili, and Wei Ren. Distributed average tracking of physical second-order agents with heterogeneous unknown nonlinear dynamics without constraint on input signals. *IEEE Transactions on Automatic Control*, 64(3):1178–1184, 2019.

[35] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of the ACM SIGCOMM*, pages 58–72, Salvador, Brazil, 2016.

[36] Martin Gregurić, Miroslav Vujić, Charalampos Alexopoulos, and Mladen Miletić. Application of deep reinforcement learning in traffic signal control: An overview and impact of open traffic data. *Applied Sciences*, 10(11):4011, 2020.

[37] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint:1507.06527*, 2015.

[38] Mark H Holmes. *Introduction to perturbation methods.* New York: Springer Science & Business Media, 2012.

[39] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *ACM SIGCOMM*, pages 15–26, Hong Kong, China, 2013.

[40] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, et al. B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan. In *Proceedings of the ACM SIGCOMM*, pages 74–87, New York, USA, 2018.

[41] Mehdi Hosseinzadeh, Emanuele Garone, and Luca Schenato. A distributed method for linear programming problems with box constraints and time-varying inequalities. *IEEE Control Systems Letters*, 3(2):404–409, 2018.

[42] Bomin Huang, Yao Zou, Ziyang Meng, and Wei Ren. Distributed time-varying convex optimization for a class of nonlinear multiagent systems. *IEEE Transactions on Automatic Control*, 65(2):801–808, 2019.

[43] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 2961–2970, Long Beach, USA, 2019.

[44] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013.

[45] Donald B Johnson. A note on dijkstra's shortest path algorithm. *Journal of the ACM*, 20(3):385–388, 1973.

[46] Maksim Kavalerov, Yuliya Likhacheva, and Yuliya Shilova. A reinforcement learning approach to network routing based on adaptive learning rates and route memory. In *Proceedings of the IEEE SoutheastCon*, pages 1–6, Charlotte, USA, 2017.

[47] Hiroki Kawamoto and Akito Igarashi. Efficient packet routing strategy in complex networks. *Physica A: Statistical Mechanics and its Applications*, 391(3):895–904, 2012.

[48] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems.* Upper Saddle River: Prentice hall, 2002.

[49] Solmaz S Kia, Jorge Cortés, and Sonia Martinez. Dynamic average consensus under limited control authority and privacy requirements. *International Journal of Robust and Nonlinear Control*, 25(13):1941–1966, 2015.

[50] Solmaz S Kia, Bryan Van Scoy, Jorge Cortes, Randy A Freeman, Kevin M Lynch, and Sonia Martinez. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. *IEEE Control Systems Magazine*, 39(3):40–72, 2019.

[51] Solmaz S Kia, Bryan Van Scoy, Jorge Cortes, Randy A Freeman, Kevin M Lynch, and Sonia Martinez. Tutorial on dynamic average consensus: the problem, its applications, and the algorithms. *IEEE Control Systems Magazine*, 39(3):40–72, 2019.

[52] Shailesh Kumar and Risto Miikkulainen. Dual reinforcement q-routing: An online adaptive routing algorithm. In *Proceedings of the International Conference on Artificial Neural Networks Engineering*, pages 231–238, Lausanne, Switzerland, 1997.

[53] Hariharan Lakshmanan and Daniela Pucci De Farias. Decentralized resource allocation in dynamic networks of agents. *SIAM Journal on Optimization*, 19(2):911–940, 2008.

[54] Sung G Lee, Yancy Diaz-Mercado, and Magnus Egerstedt. Multirobot control using time-varying density functions. *IEEE Transactions on Robotics*, 31(2):489–493, 2015.

[55] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, Hawaii, USA, 2019.

[56] Zhenhong Li, Zhengtao Ding, Junyong Sun, and Zhongkui Li. Distributed adaptive convex optimization on directed graphs via continuous-time algorithms. *IEEE Transactions on Automatic Control*, 63(5):1434–1441, 2018.

[57] Zhenhong Li, Zizhen Wu, Zhongkui Li, and Zhengtao Ding. Distributed optimal coordination for heterogeneous linear multiagent systems with event-triggered mechanisms. *IEEE Transactions on Automatic Control*, 65(4):1763–1770, 2019.

[58] Shu Liang, Xianlin Zeng, and Yiguang Hong. Distributed nonsmooth optimization with coupled inequality constraints via modified lagrangian function. *IEEE Transactions on Automatic Control*, 63(6):1753–1759, 2017.

[59] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint:1509.02971*, 2015.

[60] Hyojun Lim and Chongkwon Kim. Flooding in wireless ad hoc networks. *Computer Communications*, 24(3-4):353–363, 2001.

[61] Peng Lin, Wei Ren, and Jay A Farrell. Distributed continuous-time optimization: nonuniform gradient gains, finite-time convergence, and convex constraint set. *IEEE Transactions on Automatic Control*, 62(5):2239–2253, 2016.

[62] Antonio Mira Lopez and Douglas R Heisterkamp. Simulated annealing based hierarchical q-routing: a dynamic routing protocol. In *Proceedings of the International Conference on Information Technology: New Generations*, pages 791–796, Las Vegas, USA, 2011.

[63] Youcheng Lou, Yiguang Hong, Lihua Xie, Guodong Shi, and Karl Henrik Johansson. Nash equilibrium computation in subnetwork zero-sum games with switching communications. *IEEE Transactions on Automatic Control*, 61(10):2920–2935, 2016.

[64] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, Long Beach, USA, 2017.

[65] Jie Lu and Choon Yik Tang. Zero-gradient-sum algorithms for distributed convex optimization: The continuous-time case. *IEEE Transactions on Automatic Control*, 57(9):2348–2354, 2012.

[66] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *ACM Workshop on Hot Topics in Networks*, pages 50–56, Atlanta, USA, 2016.

[67] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 64–69, San Diego, USA, 2007.

[68] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 2520–2525. Shanghai, China, 2011.

[69] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint:1312.5602*, 2013.

[70] Daniel K Molzahn, Florian Dörfler, Henrik Sandberg, Steven H Low, Sambuddha Chakrabarti, Ross Baldick, and Javad Lavaei. A survey of distributed optimization and control algorithms for electric power systems. *IEEE Transactions on Smart Grid*, 8(6):2941–2962, 2017.

[71] Dmitry Mukhutdinov, Andrey Filchenkov, Anatoly Shalyto, and Valeriy Vyatkin. Multi-agent deep learning for simultaneous optimization for time and energy in distributed routing system. *Future Generation Computer Systems*, 94:587–600, 2019.

[72] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in Neural Information Processing Systems*, 30:2775–2785, 2017.

[73] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[74] Shahram Nosrati, Masoud Shafiee, and Mohammad Bagher Menhaj. Dynamic average consensus via nonlinear protocols. *Automatica*, 48(9):2262–2270, 2012.

[75] Reza Olfati-Saber. Distributed Kalman filtering for sensor networks. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5492–5498, New Orleans, USA, 2007.

[76] Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[77] Leonid Peshkin and Virginia Savova. Reinforcement learning for adaptive routing. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1825–1830, Hawaii, USA, 2002.

[78] James A Preiss, Wolfgang Honig, Gaurav S Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 3299–3304, Singapore, 2017.

[79] Zhirong Qiu, Shuai Liu, and Lihua Xie. Distributed constrained optimal consensus of multi-agent systems. *Automatica*, 68:209–215, 2016.

[80] Muhannad Quwaider and Subir Biswas. Dtn routing in body sensor networks with dynamic postural partitioning. *Ad Hoc Networks*, 8(8):824–841, 2010.

[81] Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *Proceedings of the International Symposium on Information Processing in Sensor Networks*, pages 20–27, 2004.

[82] Salar Rahili and Wei Ren. Distributed continuous-time convex optimization with time-varying cost functions. *IEEE Transactions on Automatic Control*, 62(4):1590–1605, 2017.

[83] Muhammad Saim, Sheida Ghapani, Wei Ren, Khalid Munawar, and Ubaid M Al-Saggaf. Distributed average tracking in multi-agent coordination: extensions and experiments. *IEEE Systems Journal*, 12(3):2428–2436, 2018.

[84] Lutz Schönemann. Evolution strategies in dynamic environments. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 51–77. Springer, 2007.

[85] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1889–1897, Lille, France, 2015.

[86] Mohamad T Shahab and Moustafa Elshafei. Distributed optimization of multi-robot motion with time-energy criterion. In *Path Planning for Autonomous Vehicles-Ensuring Reliable Driverless Navigation and Control Maneuver*. IntechOpen, 2019.

[87] Daniel Shevitz and Brad Paden. Lyapunov stability theory of nonsmooth systems. *IEEE Transactions on Automatic Control*, 39(9):1910–1914, 1994.

[88] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[89] Andrea Simonetto, Alec Koppel, Aryan Mokhtari, Geert Leus, and Alejandro Ribeiro. Decentralized prediction-correction methods for networked time-varying convex optimization. *IEEE Transactions on Automatic Control*, 62(11):5724–5738, 2017.

[90] Marcos A Simplício Jr, Paulo SLM Barreto, Cintia B Margi, and Tereza CMB Carvalho. A survey on key management mechanisms for distributed wireless sensor networks. *Computer networks*, 54(15):2591–2612, 2010.

[91] João Luís Sobrinho and Miguel Alves Ferreira. Routing on multiple optimality criteria. In *Proceedings of the ACM SIGCOMM*, pages 211–225, online, 2020.

[92] Demetri P Spanos, Reza Olfati-Saber, and Richard M Murray. Dynamic consensus on mobile networks. *IFAC World Congress*, Prague, Czech Republic, 2005, pp. 1–6.

[93] Wenjing Su. Traffic engineering and time-varying convex optimization. 2009.

[94] Devika Subramanian, Peter Druschel, and Johnny Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 832–839, Nagoya, Japan, 1997.

[95] Chao Sun, Maojiao Ye, and Guoqiang Hu. Distributed time-varying quadratic optimization for multiple agents under undirected graphs. *IEEE Transactions on Automatic Control*, 62(7):3687–3694, 2017.

[96] Shan Sun, Fei Chen, and Wei Ren. Distributed average tracking over weight-unbalanced directed graphs. In *Proceedings of the American Control Conference*, pages 1400–1405, Philadelphia, USA, 2019.

[97] Shan Sun, Fei Chen, and Wei Ren. Distributed average tracking over weight-unbalanced directed graphs. In *Proceedings of the American Control Conference*, pages 1400–1405, Philadelphia, USA, 2019.

[98] Shan Sun, Fei Chen, and Wei Ren. Distributed average tracking in weight-unbalanced directed networks. *IEEE Transactions on Automatic Control*, 2020.

[99] Shan Sun and Mariam Kiran. Multi-agent meta reinforcement learning for packet routing in dynamic network environments. 2020.

[100] Shan Sun and Wei Ren. Distributed continuous-time optimization with time-varying objective functions and inequality constraints. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5622–5627, online, 2020.

[101] Shan Sun and Wei Ren. Distributed continuous-time optimization with time-varying objective functions and inequality constraints. In *Proceedings of the IEEE Conference on Decision and Control*, pages 5622–5627, Jeju, Korea, 2020.

[102] Shan Sun, Yifan Zhang, Peng Lin, Wei Ren, and Jay A Farrell. Distributed time-varying optimization with state-dependent gains: algorithms and experiments. *IEEE Transactions on Control Systems Technology*, 2021.

[103] Andrew S Tanenbaum et al. *Computer networks.* Upper Saddle River: Prentice hall, 1996.

[104] Yujie Tang. Time-varying optimization and its application to power system operation. 2019.

[105] Nigel Tao, Jonathan Baxter, and Lex Weaver. A multi-agent, policy-gradient approach to network routing. In *Proceedings of the International Conference on Machine Learning*, pages 553–560, Williamstown, USA, 2001.

[106] Ageliki Tsioliaridou, Christos Liaskos, Eugen Dedu, and Sotiris Ioannidis. Packet routing in 3d nanonetworks: A lightweight, linear-path scheme. *Nano communication networks*, 12:63–71, 2017.

[107] Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. Learning to route. In *ACM workshop on hot topics in networks*, pages 185–191, Palo Alto, USA, 2017.

[108] Diederik Verscheure, Bram Demeulenaere, Jan Swevers, Joris De Schutter, and Moritz Diehl. Time-optimal path tracking for robots: A convex optimization approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327, 2009.

[109] Hoi-To Wai, Zhuoran Yang, Zhaoran Wang, and Mingyi Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Advances in Neural Information Processing Systems*, pages 9649–9660, Montreal, Canada, 2018.

[110] Bo Wang, Shan Sun, and Wei Ren. Distributed continuous-time algorithms for optimal resource allocation with time-varying quadratic cost functions. *IEEE Transactions on Control of Network Systems*, 7(4):1974–1984, 2020.

[111] Jing Wang and Nicola Elia. A control perspective for centralized and distributed convex optimization. In *Proceedings of the IEEE conference on decision and control and European control conference*, pages 3800–3805, Orlando, USA, 2011.

[112] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[113] Chenguang Xi, Van Sy Mai, Ran Xin, Eyad H Abed, and Usman A Khan. Linear convergence in optimization over directed graphs with row-stochastic matrices. *IEEE Transactions on Automatic Control*, 63(10):3558–3565, 2018.

[114] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. Experience-driven networking: A deep reinforcement learning based approach. In *Proceedings of the IEEE Conference on Computer Communications*, pages 1871–1879, Honolulu, USA, 2018.

[115] Shaofu Yang, Qingshan Liu, and Jun Wang. A multi-agent system with a proportional-integral protocol for distributed constrained optimization. *IEEE Transactions on Automatic Control*, 62(7):3461–3467, 2016.

[116] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019.

[117] Maojiao Ye and Guoqiang Hu. Distributed seeking of time-varying nash equilibrium for non-cooperative games. *IEEE Transactions on Automatic Control*, 60(11):3000–3005, 2015.

[118] Xinlei Yi, Xiuxian Li, Lihua Xie, and Karl H Johansson. Distributed online convex optimization with time-varying coupled inequality constraints. *IEEE Transactions on Signal Processing*, 68:731–746, 2020.

[119] Xinyu You, Xuanjie Li, Yuedong Xu, Hui Feng, Jin Zhao, and Huaicheng Yan. Toward packet routing with fully distributed multiagent deep reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[120] Huaguang Zhang, He Jiang, Yanhong Luo, and Geyang Xiao. Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method. *IEEE Transactions on Industrial Electronics*, 64(5):4091–4100, 2016.

[121] Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Networked multi-agent reinforcement learning in continuous spaces. In *Proceedings of the International Conference on Decision and Control*, pages 2771–2776, Miami Beach, USA, 2018.

[122] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *Proceedings of the International Conference on Machine Learning*, pages 5872–5881, Stockholm, Sweden, 2018.

[123] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1701–1709, Beijing, China, 2014.

[124] Yanqiong Zhang, Zhenhua Deng, and Yiguang Hong. Distributed optimal coordination for multiple heterogeneous Euler-Lagrangian systems. *Automatica*, 79:207–213, 2017.

[125] Yu Zhao, Yongfang Liu, Zhongkui Li, and Zhisheng Duan. Distributed average tracking for multiple signals generated by linear dynamical systems: an edge-based framework. *Automatica*, 75:158–166, 2017.

[126] Yu Zhao, Yongfang Liu, Guanghui Wen, and Tingwen Huang. Finite-time distributed average tracking for second-order nonlinear systems. *IEEE Transactions on Neural Networks and Learning systems*, 30(6):1780–1789, 2019.

[127] Yu Zhao, Yongfang Liu, Guanghui Wen, Xinghuo Yu, and Guanrong Chen. Distributed average tracking for lipschitz-type of nonlinear dynamical systems. *IEEE Transactions on Cybernetics*, 49(12):4140–4152, 2019.

[128] Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.

[129] Yanan Zhu, Wenwu Yu, Guanghui Wen, and Wei Ren. Continuous-time coordination algorithm for distributed convex optimization over weight-unbalanced directed networks. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 66(7):1202–1206, 2019.

.1 Appendix

Here, we provide the derivation of Equation (5.2). All the notations carry the same meaning as those in Section 5.2.1. The probability of a trajectory $\tau = (s_0, a_0, \cdots, s_H, a_H, S_{H+1})$ given that actions come from π_θ is

$$\begin{aligned} P(\tau|\pi_\theta) &= \rho(s_0)\Pi_{t=0}^{H} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t) \\ &= \rho(s_0)\Pi_{t=0}^{H} P(s_{t+1}|s_t, a_t)\Pi_{i=1}^{n}\pi_{\theta^i}^{i}(a_t^i|o_t^i). \end{aligned}$$

The log-probability of a trajectory is

$$\begin{aligned} \log P(\tau|\pi_\theta) &= \log\rho(s_0) + \sum_{t=0}^{H}\log\left[P(s_{t+1}|s_t, a_t)\Pi_{i=1}^{n}\pi_{\theta^i}^{i}(a_t^i|o_t^i)\right] \\ &= \log\rho(s_0) + \sum_{t=0}^{H}\log\left[P(s_{t+1}|s_t, a_t)\right] \\ &\quad + \sum_{t=0}^{H}\sum_{i=1}^{n}\log\left[\pi_{\theta^i}^{i}(a_t^i|o_t^i)\right]. \end{aligned}$$

The gradient of the log-probability of a trajectory is

$$\begin{aligned}\nabla_\theta \log P(\tau|\pi_\theta) &= \nabla_\theta \log \rho(s_0) + \sum_{t=0}^{H} \nabla_\theta \log \left[P(s_{t+1}|s_t, a_t)\right] \\ &\quad + \sum_{t=0}^{H}\sum_{i=1}^{n} \nabla_\theta \log \left[\pi^i_{\theta^i}(a^i_t|o^i_t)\right] \\ &= \sum_{t=0}^{H}\sum_{i=1}^{n} \nabla_\theta \log \pi^i_{\theta^i}(a^i_t|o^i_t),\end{aligned}$$

and thus

$$\nabla_{\theta^i} \log P(\tau|\pi_\theta) = \sum_{t=0}^{H} \nabla_{\theta^i} \log \pi^i_{\theta^i}(a^i_t|o^i_t).$$

Putting the above equations together, we have the following

$$\begin{aligned}\nabla_{\theta^i} J(\theta) &= \nabla_{\theta^i} \mathbb{E}[R(\tau)] \\ &= \nabla_{\theta^i} \int_\tau P(\tau|\pi_\theta)R(\tau) = \int_\tau \nabla_{\theta^i} P(\tau|\pi_\theta)R(\tau) \\ &= \int_\tau P(\tau|\pi_\theta)\nabla_{\theta^i} \log P(\tau|\pi_\theta)R(\tau) \\ &= \mathbb{E}\left[\nabla_{\theta^i} \log P(\tau|\pi_\theta)R(\tau)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{H} \nabla_{\theta^i} \log \pi^i_{\theta^i}(a^i_t|o^i_t)R(\tau)\right].\end{aligned}$$