COMMUNICATION AND COMPUTATION IN LARGE-SCALE DISTRIBUTED OPTIMIZATION

by

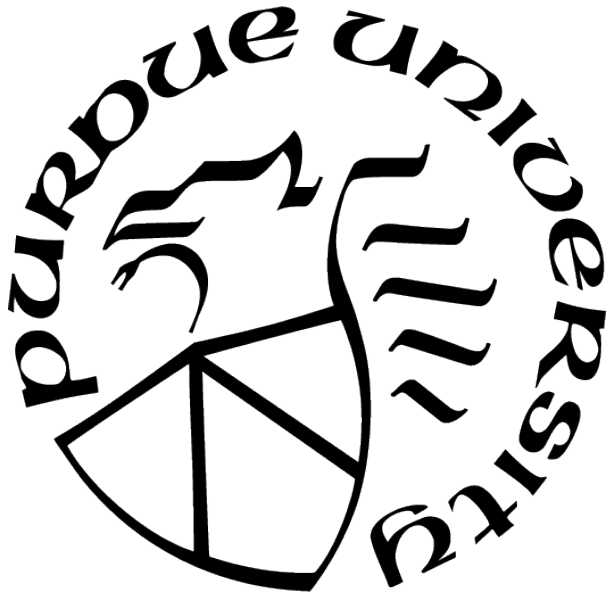
Bin Du

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Aeronautics and Astronautics

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Dengfeng Sun, Chair

School of Aeronautics and Astronautics

Dr. Inseok Hwang

School of Aeronautics and Astronautics

Dr. Jianghai Hu

School of Electrical and Computer Engineering

Dr. Nan Kong

Weldon School of Biomedical Engineering

Approved by:

Dr. Gregory A. Blaisdell

Dedicated to my parents and my sister.

ACKNOWLEDGMENTS

Perhaps, the first sentence of almost all thesis acknowledgment goes like: "First of all, I would like to express my deepest gratitude to my advisor, Dr. xxx", I was trying to say something different, but failed to find a single thing in my mind that is more important. I could never forget the moment when you adopted me as an AAE student in 2018. Since then, your unstopped support and patience have helped me a lot, not only in the research but also in my daily life. I really appreciate every advice you have provided for me and every discussion we have had together. People keep telling that doing a Ph.D. is never easy, but I say that you have made my Ph.D. journey in AAE incredibly enjoyable. So, thank you, Dr. Dengfeng Sun, for everything you have done for me.

Many thanks go to my Ph.D. dissertation committee members, Prof. Inseok Hwang, Prof. Jianghai Hu, and Prof. Nan Kong. Without their guidance and insightful comments on my work, this dissertation would not have been possible. In addition, I am also grateful for the collaboration with Dr. Chris Claudel and Mr. Kun Qian from the University of Texas at Austin, as well as Dr. Jun Chen from San Diego State University. The discussions we have had are really delightful and helpful for the research. Deep gratitude also goes to my lab mates and friends: Jiazhen Zhou, Dawei Sun, Xuan Wang, Wanxin Jin, Hanyao Hu, Lian Yuan, Chan-Yuan Kuo, You-Ru Lu, Shujin Jiang, Ruijiu Mao. Thank you all, for the joyful moments we have experienced together.

Last but not least, I am deeply indebted to my parents and my sister. Studying abroad might be one of the most beneficial experiences for me, but certainly not for them. There are not enough words to express my gratitude for their endless love and supporting me to pursue my dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of positive real numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{N}_+$	set of positive natural numbers
$\mathbf{I}_p$	$p \times p$ identity matrix
$\mathbf{1}_I$	I-dimensional one vector with all elements being 1
$\mathbf{e}_i$	unit vector, i-th column of the identity matrix
$\mathcal{I}_i(\cdot)$	i-th element of the input vector
$\|\cdot\|$	$\mathcal{L}_2$ vector norm
$\otimes$	kronecker product
$\text{diag}\{\cdots\}$	diagonal matrix
$(\cdot)^k$	k-th power of the input matrix/scalar
$\nabla f(\cdot)$	gradient of the function $f(\cdot)$
$\nabla^2 f(\cdot)$	Hessian matrix of the function $f(\cdot)$
$\text{dist}(\cdot,\cdot)$	Euclidean distance between two spaces
$\text{span}\{\cdot\}$	space spanned by the input vector
$\text{trace}(\cdot)$	trace of the input matrix
$\det(\cdot)$	determinant of the input matrix
$\langle\cdot,\cdot\rangle$	inner product of the input vectors

ABBREVIATIONS

UAV	unmanned aerial vehicle
DGD	distributed gradient descent
DGDx	improved distributed gradient descent with inexact average consensus
NetProx	generalized networked proximal
PH	progressive hedging
DistPH	distributed progressive hedging
CPU	central processing unit
TCP/IP	transmission control protocol/Internet protocol
ADMM	alternating direction method of multipliers
SGD	stochastic gradient descent
UCB	confidence upper bound
D-UCB	dummy confidence upper bound

ABSTRACT

Last decades have witnessed a surge of interest in developing distributed algorithms for solving large-scale optimization problems with multi-agent systems. The development of a standard distributed optimization algorithm can be typically summarized as two fundamental stages – communication and computation. Whereas the communication stage enables agents to collect the desired information from the entire network, the computation stage updates each agent's local state towards the optimizer of the objective function. This dissertation focuses on the two key aspects of distributed optimization, develops a suite of new provable distributed algorithms for solving both deterministic and stochastic optimization problems, and further explores the opportunities of applying the distributed optimization techniques into real-world applications especially with multi-UAV systems. First, starting from the perspective of communication, a DGDx algorithm is proposed by incorporating multiple communication iterations into the consensus step. Second, observing that the standard DGD algorithm is a special case where the linear approximation is applied for the local objective functions, we develop a NetProx algorithm which generalizes the choice of approximation function for each agent and thus provides flexibility in the aspect of computation. The two proposed algorithms both offer great potential to balance the communication and computation in distributed optimization. In addition, we consider the issue of network communication attacks and devise a resilient algorithm for solving the specific distributed min-max problem. Two-stage stochastic programs are also studied and a DistPH algorithm is developed by adapting the classical PH method under a peer-to-peer multi-agent network. At last, two real-wold applications – distributed data fusion and distributed source tracking, both involved with the multi-UAV system, are investigated to demonstrate the superiority of the distributed optimization techniques. Theoretical analysis and numerical results are provided to validate the effectiveness of all proposed algorithms.

1. INTRODUCTION

1.1 Background and Motivations

The recent and still ongoing explosion in the size and complexity of datasets, as well as the increasing demand of computational resources, have driven us to what is oftentimes called the big-data era. Meanwhile, the present computational performance of one single machine is, however, far from sufficient to meet such stringent computational requirements. As a direct consequence, there is an undoubted necessity of developing new tools and paradigms to enhance the computing capabilities of the current machines. To this end, a natural idea is, instead of focusing on the individual machines, but to construct a cooperative networked architecture which is oftentimes referred to as the multi-agent system in the literature. It is expected that, through the cooperation among the multiple agents, the overall computational workload is spread over the entire networked system and thus is reduced for each single agent within the network. In fact, such an idea has already given rise to a remarkable shift in focus of solving large-scale problems in the control and optimization community over the last few decades. Much effort has been devoted to the design of distributed algorithms, rather than the conventional centralized ones, which are more favorable to exploit the special structure of the multi-agent computing environment. At the same time, these distributed algorithms have also been widely adopted in various real-world applications, including but not limited to, wireless sensor network [1]–[3], transportation systems [4]–[6], distributed machine learning [7]–[9], power system control [10]–[12], etc.

It can be seen from the above introduction that two key aspects of the distributed processing can be attributed to communication and computation. Whereas the communication inherently enables each individual agent to take advantage of the computing capability of the entire network, the computation drives all agents to approach the states where they desire. Corresponding to these two aspects, the development of distributed algorithms can be also summarized as two stages: i) a communication stage, i.e., to determine what kind of information the agents should communicate with each other; and ii) a computation stage, i.e., to devise how to update each agent's local state based on the information maintained by itself and also received from its immediate neighbors. Indeed, the majority of existing algorithms

in the literature are designed according to these two stages. Nevertheless, when it comes to the evaluation of the distributed algorithms, it seems that researchers are overly interested in the computation side, but neglect the efficiency involved with the communication part. One good example here is that the convergence rate of algorithm is typically characterized with respect to the number of iterations, and in fact, each iteration is corresponding to one step of computation that is simultaneously performed by all agents within the network. In other words, in such a way of evaluation, the amount of communication cost is overlooked or simply counted equally with the computation cost at each iteration. However, those two types of cost can be significantly different from each other in some specific scenarios. For instance, the recent study [13] suggests that CPUs we are using nowadays can read and write the memory at over 10 – 100GB per second, whereas the speed of communication over TCP/IP is about 100MB per second. This implies that the gap between the cost of intra-node computation and inter-node communication can be at worst three orders of magnitude. On this account, we will have to rethink the evaluation and even the development of distributed algorithms. In short, a promising distributed algorithmic framework should take into account both communication and computation aspects, and further have the flexibility to balance the two types of cost for different applications.

In addition, another underlying assumption that is made by most of the existing distributed algorithms is the perfection of communication. That is, all individual agents within the network are able to instantaneously transmit their local information without any error to the targets. In reality, however, one can barely assume that the information transmissions are always perfect, due to many factors such as communication delay [14], [15], data quantization [16], [17], packet loss [18], [19] and communication attack [20], [21]. Among those various imperfections, the communication attack is often regarded as the most challenging one and has attracted a significant amount of research attention, since the attacked communication channel may not only lose its functionality, but also be able to inject malicious information into the whole system. In this case, it is of great importance that the distributed algorithms are devised in such a way that they can tolerate the adversarial behaviors to some extent and still ensure the effectiveness of the system with a guaranteed

performance. Therefore, an appealing distributed algorithm should be also resilient against the potential network communication attacks.

Motivated by the aforementioned issues, this dissertation focuses on the two key aspects of the distributed optimization – communication and computation, develops a suite of new provable distributed algorithms for solving both deterministic and stochastic optimization problems, and further explores the opportunities of applying the idea of distributed optimization into real-world applications especially with multi-UAV systems.

1.2 Literature Review

Indeed, the distributed optimization problem has gained increasingly significant attention over the last few decades, and it has found a large variety of applications in diverse areas, including networked system control [22], [23], large-scale signal processing [24], [25], machine learning [7], [8], [26], just to name a few. In order to solve the optimization problem over multi-agent networks, there have been a significant amount research in the context of distributed algorithms.

For the distributed deterministic optimization, the pioneering work can be traced back to Bertsekas and Tsitsiklis [27], [28] in 1980s. After that, with the advance of communication technology as well as the emergence of big-data research, more and more effort has been devoted into the development of distributed algorithms. A seminal work [16], published in 2009, proposed the famous distributed (sub-)gradient descent (DGD) algorithm which integrates the basic idea of consensus with the gradient descent method in the conventional centralized optimization. Although it makes great progress in the study of distributed optimization, a notable convergence result [16], [29] of the DGD algorithm is that, when the constant step-size α is adopted, it cannot converge to the exact optimal solution, but only gives an inexact one which is in the optimal solution's $\mathcal{O}(\alpha)$-neighborhood. Moreover, the authors in [29] proved that the rate of convergence to the $\mathcal{O}(\alpha)$-neighborhood is in the order of $\mathcal{O}(1/k)$ (k is the iteration index) when the objective function is convex, and the convergence rate will be upgraded to exponential when the objective function is strongly convex. In addition, further extensions of the DGD algorithm are also investigated in the subsequent

works; for example, the projected DGD algorithm is proposed in [30] which deals with the constrained distributed optimization problems, and Nesterov's acceleration is adopted in [31] to develop the so-called fast DGD algorithm.

Note that a typical setting of the distributed optimization assumes that each agent privately has access to its own local objective function and thus only local gradient is available when performing the gradient descent steps. On this account, many works attributed the inexact convergence of the DGD algorithm to the lack of global gradients for the individual agents; see a detailed explanation in Section 2.1 in Chapter 2. This observation has motivated two main paths to further improve the convergence result of the DGD algorithm. First, the authors in [31]–[33] show that the exact convergence can be obtained by adopting a diminishing step-size rule with the standard DGD algorithm. The guideline is to diminish the error caused by the using of local gradients. However, the convergence rate is significantly compromised compared with the centralized gradient descent method due to the diminishing step-size. The second line of works focuses on the adjustment of local gradients. An idea of gradient tracking is applied in [34]–[36] to substitute the local gradients with a tracked value of the global gradient. The methods, proposed in [37]–[39], introduce a correction of the local gradient by combining the gradient information obtained from the last two consecutive iterations. As a result, a linear convergence rate, which is comparable to the centralized case, can be achieved when the objective function is smooth and strongly convex. In addition, the analysis in [40] further points out that the aforementioned two types of methods can be unified from a primal-dual perspective. It is also worth mentioning that there are other distributed algorithms which achieve the linear convergence rate; see e.g., [41]–[45]. The authors in [41]–[43] proved the linear convergence rate of distributed alternating direction method of multipliers (ADMM). The second-order methods are also proposed in [44], [45], which also obtained the linear convergence rate.

Whereas deterministic optimization problems are formulated with all known parameters, real-world problems almost invariably include parameters which are subject to some unknown uncertainties. Under such circumstances, stochastic optimization, as an powerful approach for modeling and further solving the problems involved with uncertainties, has

found great potential of applications in the area of data analytics [13], [46]–[49], especially in the statistical learning theory [50]–[52] recently.

In order solve the stochastic optimization problems, dominant approaches are based on the idea of stochastic approximation, also known as stochastic gradient descent (SGD) method, which was initially proposed in [53] in 1950s. The classic SGD method mimics the conventional centralized gradient descent by replacing the full gradient with its unbiased estimation. Since then, the SGD method has attracted much interest [54]–[58]. To solve the problems with a networked architecture, the mini-batch SGD method [59], [60] provides a straightforward way to parallelize the procedure of SGD. Even though the standard SGD method can achieve the optimality by using the estimation of gradient, due to the variance of estimator, yet it usually obtains slow convergence and poor performance [61]. Recently, with the help of the so-called variance reduction technique, various stochastic variance reduced methods [61]–[64] and their proximal variants [65]–[67] are proposed in the literature. The parallel versions of these methods are also investigated in [68]–[70]. All of these variance reduction based methods achieve remarkable linear convergence rate which is comparable to the conventional full gradient descent method. However, it should be noted that most of the parallel methods can only be implemented with a master-worker network architecture in which a mater node is always present to provide central coordinations for all worker. In contrast to such a master-worker architecture, the peer-to-peer multi-agent network is often preferred due to the following reasons: i) communication burden for the master node could be extremely heavy or even prohibited, as it is required to provide the direct communication channel for each of the workers; ii) the cyber-security concern may arise, since failure of the master node can lead to the collapse of whole system; and iii) the privacy of node information cannot be preserved in the master node, as it collects all information in the entire network. Furthermore, it is worthy noting that the master-worker architecture could be a special case of the peer-to-peer multi-agent network, since the central node can serve as a master when the considered network has a special star topology.

In this dissertation, we focus on a famous and widely-studied special instance of the stochastic optimization problem – two-stage stochastic programs. In fact, such two-stage stochastic programs have found numerous applications in energy planning [71], manufactur-

ing [72], logistic [73], etc. To deal with this stochastic program, a classical methodology builds on the idea of representing the uncertainty with a finite number of scenarios, and then various decomposition-based methods are proposed, including the Bender's decomposition [74]–[76] (also known as the L-shaped method [77]), dual decomposition [78], [79], among others [80]–[82]. The Bender's decomposition basically separates the original problem into a master problem and a group of subproblems for each scenario, and then solves these two types of problems iteratively. The solutions of subproblems help to further shape constraints of the master problem by adding optimality or feasibility cuts. In the dual decomposition, the original problem is decomposed directly into a set of scenario-based subproblem. Subsequently, a well-known progressive hedging (PH) method [78] is developed to solve for the solutions. It turns out that both two decomposition methods can be applied to efficiently solve the two-stage stochastic program, especially when the second-stage problem has a large number of scenarios. Another benefit of them is attributed to their straightforward parallelization [83], [84]. However, as emphasized previously, the parallel architecture always assumes a master node, and the communication cost could be extremely heavy or even prohibited. Moreover, the master problem derived by Bender's decomposition often has much larger size than the subproblems. Thus, the computation cost for the master node is also an issue in this case.

1.3 Dissertation Overview and Contributions

As indicated previously, this dissertation aims to develop a suite of new distributed algorithms by focusing on the two key aspects of the distributed optimization, i.e., communication and computation. Both deterministic and stochastic optimization problems are studied in this work. Particularly, we are interested in devising the algorithmic frameworks which are capable of providing flexibilitiy to balance the communication and computation cost, and also being resilient against the potential network communication attacks. In addition, we also explore the opportunities of applying the idea of distributed optimization into real-world applications especially with multi-UAV systems. The detailed contributions of each chapter is summarized as follows.

In Chapter 2, a novel DGDx algorithm is proposed, focusing on the communication aspect and aiming at improving the inexact convergence of the standard DGD algorithm. It is proved that the exact convergence can be achieved when the introduced sequence of consensus error bounds decays to zero; furthermore, it converges at a linear rate when the sequence decays to zero linearly. Provided the flexibility of choices of both consensus error bounds and consensus schemes, our DGDx algorithm offers the potential to balance the communication and computation cost for solving the distributed optimization problem. Furthermore, a novel average consensus scheme is also presented which only involves the row-stochastic weight matrices, as distinct from the classical weighted averaging scheme which requires the weight matrix to be doubly-stochastic. Such an average consensus scheme provides an important building block for the implementation of algorithms developed in the subsequent chapters.

Chapter 3 alternatively starts from the computation aspect of the distributed optimization and develops the generalized NetProx framework. It is shown that the standard DGD algorithm can be viewed as a special case of our NetProx framework by adopting the linear approximation of the local objective functions. Such approximation functions are then generalized into any form as long as the given conditions are satisfied. In addition, we prove that, with the higher order approximation functions, the convergence of the NetProx algorithm will be accelerated to some extent, but meanwhile at the price of more computational cost for each agent at each iteration. In this sense, the flexibility of choices of the approximation functions also helps to balance the computation and communication in distributed optimization.

In Chapter 4, we further consider the issue of network communication attacks. A promising resilient algorithm is proposed to solve the distributed min-max optimization, which is a special instance of the general distributed optimization problem. One major contribution of our resilient algorithm is that we aim to obtain the exact global optimal solution, which takes into account the local objective functions from all agents no matter whose communication channels are attacked or non-attacked. Such a goal is quite challenging or even impossible for the existing resilient distributed algorithms. We prove that this goal can be achieved un-

der some reasonable and arguably necessary assumptions, e.g., the attacked communication channels can be recovered within a certain time-window.

Chapter 5 studies the stochastic optimization problems. More specifically, we adapt the classical PH method under the peer-to-peer multi-agent network and propose the DistPH algorithm for solving two-stage stochastic programs. Theoretical convergence guarantee for the DistPH algorithm is proved when the first- and second-stage decision variables are both continuous and subject to convex constraints. In addition, the initialization and termination verification mechanisms are designed in the distributed setting. We further adapt the DistPH algorithm to the Lagrangian dual lower-bound computation for two-stage stochastic programs with mixed-integer variables. The benefits of the proposed distributed algorithm in both aspects of communication and computation are verified through comprehensive simulation studies on benchmark stochastic programming instances.

In Chapter 6, two real-world applications involved with the multi-UAV systems are investigated to confirm the superiority of the distributed optimization in terms of communication and computation against the centralized one. The first application concerns a health-care scenario where the health status of a certain target in rural environment needs to be monitored by the multi-UAV system through sensing some vital on-scene biological signals. The second one studies a real-world methane leaking source tracking mission by the multi-UAV system operating in an unknown and dynamical environment. Applying the average/sum consensus scheme developed in the previous chapters, a distributed data fusion algorithm and a distributed on-line source tracking algorithm are developed to solve the two problems, respectively. Theoretical findings are all validated via the numerical results on both applications.

At last, conclusions and future directions that could extend the work in this dissertation are provided in Chapter 7.

1.4 Preliminaries and Assumptions

1.4.1 Problem Formulation

This dissertation generally focuses on solving the following minimization problem, involving a network of I agents,

$$\min_{\mathbf{x}\in\mathcal{X}} \quad F(\mathbf{x}) := \sum_{i=1}^{I} f_i(\mathbf{x}), \tag{P}$$

over the common global decision variable $\mathbf{x} \in \mathbb{R}^p$ subject to the feasible set $\mathcal{X} \subseteq \mathbb{R}^p$. In this Problem (P), each agent $i \in \mathcal{I} := \{1, 2, \cdots, I\}$ is assumed to privately own a local objective function $f_i(\cdot) : \mathbb{R}^p \to \mathbb{R}$ and be capable of exchanging information with its immediate neighbors. Throughout this dissertation, we assume that the set of minimizers of the global objective function $F(\cdot) : \mathbb{R}^p \to \mathbb{R}$ is not empty, and study Problem (P) under the following underlying assumptions.

- **On the feasible set $\mathcal{X}$**

Assumption 1.4.1. *The feasible set $\mathcal{X} \neq \emptyset$ is assumed to be closed and convex.*

In some special cases, we incorporate the boundedness into the set $\mathcal{X}$, i.e.,

Assumption 1.4.2. *The feasible set $\mathcal{X} \neq \emptyset$ is assumed to be compact and convex.*

- **On the objective function $F(\cdot)$**

Assumption 1.4.3. *Each local objective function $f_i(\cdot)$ is assumed to be differentiable and its gradient ∇f_i is L_i^f-Lipschitz continuous on the feasible set $\mathcal{X}$, i.e., there exists a constant $L_i^f > 0$ such that,*

$$\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L_i^f \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{1.1}$$

Note that, under Assumption 1.4.2, the gradient ∇f_i on each point $\mathbf{x} \in \mathcal{X}$ must be bounded due to the compactness of the feasible set $\mathcal{X}$, i.e., there exists a constant $\kappa > 0$ such that

$$\|\nabla f_i(\mathbf{x})\| \leq \kappa, \quad \forall \mathbf{x} \in \mathcal{X}, \ \forall i \in \mathcal{I}. \tag{1.2}$$

Assumption 1.4.4. *Each local objective function $f_i(\cdot)$ is assumed to be convex on the feasible set $\mathcal{X}$, i.e.,*

$$f_i(\mathbf{y}) \geq f_i(\mathbf{x}) + \nabla f_i(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}), \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{1.3}$$

In some special cases, we will also need the strong-convexity of the the global objective function $F(\cdot)$, which is stated as follows.

Assumption 1.4.5. *The global objective function $F(\cdot)$ is assumed to be μ-strongly convex on the feasible set $\mathcal{X}$, i.e., there exists a constant $\mu > 0$ such that*

$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \geq \mu \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{1.4}$$

- **On the topology of multi-agent network**

We consider a generic time-varying directed graph for the multi-agent network, denoted as $\mathcal{G}^k := (\mathcal{I}, \mathcal{E}^k)$ at each discrete time-step k, where $\mathcal{I} = \{1, 2, \cdots, I\}$ represents the set of I nodes and $\mathcal{E}^k \subseteq \mathcal{I} \times \mathcal{I}$ represents the set of directed edges. In particular, we denote an edge as $(i, j) \in \mathcal{E}^k$ if the node i can receive information from the node j at the time-step k; by convention, we consider each node's self-loop as an edge, i.e., $(i, i) \in \mathcal{E}^k$, $\forall i \in \mathcal{I}, k \in \mathbb{N}_+$. Note that the graph $\mathcal{G}^k$ can be also described by the adjacency matrix $A^k = [a_{ij}^k]_{i,j=1}^I \in \mathbb{R}^{I \times I}$, where $a_{ij}^k = 1$ if $(i, j) \in \mathcal{E}^k$ and $a_{ij}^k = 0$ otherwise. In addition, we use a set $\mathcal{N}_{i,\text{in}}^k$ to denote the in-neighborhood of the node i at the time-step k, i.e., $\mathcal{N}_{i,\text{in}}^k := \{j \mid (i, j) \in \mathcal{E}^k\}$, and similarly, use $\mathcal{N}_{i,\text{out}}^k$ to denote its out-neighborhood, i.e., $\mathcal{N}_{i,\text{out}}^k := \{j \mid (j, i) \in \mathcal{E}^k\}$. As a consequence, $d_{i,\text{in}}^k := |\mathcal{N}_{i,\text{in}}^k|$ and $d_{i,\text{out}}^k := |\mathcal{N}_{i,\text{out}}^k|$ are used to denoted the in-degree and out-degree, i.e., the number of in- and out-neighbors, respectively. All the in- and out-degrees are combined as the degree matrices, i.e., $D_{\text{in}}^k := \text{diag}\{d_{1,\text{in}}^k, d_{2,\text{in}}^k, \cdot, d_{I,\text{in}}^k\} \in \mathbb{R}^{I \times I}$ and $D_{\text{out}}^k := \text{diag}\{d_{1,\text{out}}^k, d_{2,\text{out}}^k, \cdot, d_{I,\text{out}}^k\} \in \mathbb{R}^{I \times I}$. Note that, in some special cases where we consider the graph $\mathcal{G}^k$ to be undirected, it holds that $\mathcal{N}_{i,\text{in}}^k = \mathcal{N}_{i,\text{out}}^k$ and thus $d_{i,\text{in}}^k = d_{i,\text{out}}^k$. In these cases, we will simply use $\mathcal{N}_i^k$ and d_i^k to denote the neighborhood and degree, respectively; as a result, the degree matrix will become $D^k := \text{diag}\{d_1^k, d_2^k, \cdot, d_I^k\} \in \mathbb{R}^{I \times I}$ Moreover, if the

considered multi-agent network is time-invariant, we will further omit the superscript k, i.e., using $\mathcal{G} := (\mathcal{I}, \mathcal{E})$ to denote the underlying graph.

The following assumptions regarding the topology of the multi-agent network will be needed.

Assumption 1.4.6. *For the generic time-varying directed graph $\mathcal{G}^k = (\mathcal{I}, \mathcal{E}^k)$, it is assumed to be B-strongly connected, i.e., there exists a positive integer $B > 0$ such that the joint graph $\cup_{t=k}^{B+k-1}\mathcal{G}^t = (\mathcal{I}, \cup_{t=k}^{B+k-1}\mathcal{E}^t)$, $\forall k \in \mathbb{N}_+$ is strongly-connected.*

In particular, when the considered network is directed but time-invariant, we have the following assumption.

Assumption 1.4.7. *For the time-invariant directed graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, it is assumed to be strongly connected, i.e., there must exist an unidirectional path from every node i to every other node j within the graph.*

Further, we also consider the special case when the network is time-invariant and undirected.

Assumption 1.4.8. *For the time-invariant undirected graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$, it is assumed to be connected, i.e., there must exist a bidirectional path from every node i to every other node j within the graph.*

Compliant with the underlying multi-agent network $\mathcal{G}^k$, we next introduce the weight matrix $W^k = [w_{ij}^k]_{i,j=1}^I \in \mathbb{R}^{I\times I}$ at each time-step k, satisfying the following conditions.

Assumption 1.4.9. *Given the generic time-varying directed graphs $\mathcal{G}^k = (\mathcal{I}, \mathcal{E}^k)$, there exists a constant $\omega > 0$ such that the weight matrices W^k has i) $w_{ij}^k \geq \omega$ if $(i, j) \in \mathcal{E}^k$; and ii) $w_{ij}^k = 0$ otherwise.*

There are also three scenarios in terms of the stochasticity of the weight matrix W^k.

Assumption 1.4.10. *Each of the weight matrices W^k is row-stochastic, i.e., $W^k\mathbf{1}_I = \mathbf{1}_I$.*

Assumption 1.4.11. *Each of the weight matrices W^k is column-stochastic, i.e., $\mathbf{1}_I^\top W^k = \mathbf{1}_I^\top$.*

Assumption 1.4.12. *Each of the weight matrices W^k is doubly-stochastic, i.e., $W^k\mathbf{1}_I = \mathbf{1}_I$ and $\mathbf{1}_I^\top W^k = \mathbf{1}_I^\top$.*

1.4.2 Distributed Gradient Descent Algorithm

As we have reviewed in the last section, a seminal work [16] proposes the well-known DGD algorithm for solving Problem (P). In order to facilitate our discussion in the following chapters, let us here introduce the DGD algorithm in a relatively detailed manner. The DGD algorithm, which integrates the weighted averaging consensus scheme and the conventional gradient descent method, performs the following iterations,

$$\mathbf{x}_i^{k+1} = \sum_{j\in\mathcal{N}_i} w_{ij}\mathbf{x}_j^k - \alpha^k \nabla f_i(\mathbf{x}_i^k), \quad \forall i = \mathcal{I}, \tag{1.5}$$

where $\mathbf{x}_i^k \in \mathbb{R}^p$ is the state maintained by the i-th agent locally at the time-step k; $\{\alpha^k\}_{k\in\mathbb{N}_+}$ is a sequence of step-sizes. Note that here some underlying assumptions are incorporated: i) the multi-agent network follows a time-invariant undirected graph $\mathcal{G}$ and is assumed to be connected (see Assumption 1.4.8); ii) the weight matrix $W = [w_{ij}]_{i,j=1}^n$ is compliant with graph $\mathcal{G}$ (see Assumption 1.4.9) and assumed to be doubly-stochastic 1.4.12; and iii) the feasible set is assumed to be $\mathcal{X} = \mathbb{R}^p$, otherwise a projection operation will be needed.

In order to represent the DGD algorithm in a compact form, let us denote $\mathbf{x}^k \in \mathbb{R}^{Ip}$ as the aggregates of local states $\mathbf{x}_i^k$'s, i.e.,

$$\mathbf{x}^k := \left[(\mathbf{x}_1^k)^\top, (\mathbf{x}_2^k)^\top, \cdots, (\mathbf{x}_I^k)^\top\right]^\top. \tag{1.6}$$

Similarly, we denote $\nabla\mathbf{f}(\mathbf{x}^k) \in \mathbb{R}^{Ip}$ as the aggregates of local gradients $\nabla f_i(\mathbf{x}_i^k)$'s, i.e.,

$$\nabla\mathbf{f}(\mathbf{x}^k) := \left[\nabla f_1(\mathbf{x}_1^k)^\top, \nabla f_2(\mathbf{x}_2^k)^\top, \cdots, \nabla f_I(\mathbf{x}_I^k)^\top\right]^\top. \tag{1.7}$$

Thus, the DGD algorithm (1.5) can be equivalently expressed as

$$\mathbf{x}^{k+1} = \mathbf{W}\mathbf{x}^k - \alpha^k \nabla\mathbf{f}(\mathbf{x}^k), \tag{1.8}$$

where the matrix $\mathbf{W}$ is defined as $\mathbf{W} := W \otimes \mathbf{I}_p \in \mathbb{R}^{Ip\times Ip}$ with $\mathbf{I}_p$ being the $p \times p$ identity matrix.

1.4.3 Technical Lemmas

In this subsection, we introduce some technical lemmas that will be frequently used for establishing the convergence and analyzing the properties of our algorithms proposed in the following chapters. Note that here the detailed proofs will be omitted and the interested readers will be referred to the corresponding papers.

The first lemma establishes the crucial consensus result when the underlying graph is (strongly-)connected and the compliant weight matrix W is row-stochastic.

Lemma 1.4.1 (Proposition 1 in [85]). *Suppose that the weight matrix W satisfies Assumptions 1.4.9 and 1.4.10 and the underlying graph satisfies Assumption 1.4.8 or 1.4.7, then there must exist two constants $c_0 > 0$ and $0 < \rho_0 < 1$ such that*[1]

$$\|(W)^k - \mathbf{1}_I \boldsymbol{\pi}^\top\| \leq c_0 \cdot (\rho_0)^k, \tag{1.9}$$

where $\boldsymbol{\pi} \in \mathbb{R}^I$ is the normalized left eigenvector of the matrix W which corresponds to the eigenvalue 1, i.e., $\mathbf{1}_I^\top \boldsymbol{\pi} = 1$ and $\boldsymbol{\pi}^\top W = \boldsymbol{\pi}^\top$.

We shall remark that the above Lemma 1.4.1 inherently guarantees the asymptotic consensus for the following famous weighted averaging consensus scheme as k goes to infinity,

$$\mathbf{x}^{k+1} = \mathbf{W}\mathbf{x}^k. \tag{1.10}$$

In particular, if the weight matrix is further doubly-stochastic; see Assumption 1.4.12, then the convergence result will be the average consensus, i.e., $\lim_{k\to\infty} \|(W)^k - (1/I) \cdot \mathbf{1}_I \mathbf{1}_I^\top\| = 0$.

The subsequent two supporting lemmas state the convergence of the convolution of two positive scalar sequences.

Lemma 1.4.2 (Lemma 7 in [30]). *Given a constant $0 < \lambda < 1$ and a positive scalar sequence $\{\beta^k\}_{k\in\mathbb{N}_+}$ which has $\lim_{k\to\infty} \beta^k = 0$, then it holds that,*

$$\lim_{k\to\infty} \sum_{t=0}^{k} (\lambda)^{k-t} \beta^t = 0. \tag{1.11}$$

[1] Throughout this dissertation, we use $(\cdot)^k$ to denote the k-th power of the input scalar/matrix.

Lemma 1.4.3 (Lemma 7 in [30]). *Given a constant* $0 < \lambda < 1$ *and a positive scalar sequence* $\{\beta^k\}_{k\in\mathbb{N}_+}$ *which is assumed to be summable, i.e.,* $\sum_{k=0}^{\infty}\beta^k < \infty$*, then it holds that,*

$$\lim_{k\to\infty}\sum_{t=0}^{k}\sum_{s=0}^{t}(\lambda)^{t-s}\beta^s < \infty. \tag{1.12}$$

The last lemma, stating the super-martingale convergence result, will also play a crucial role in the proofs of our own lemmas and theorems.

Lemma 1.4.4 (Lemma 1 in [86]). *Let* $\{a^k\}_{k\in\mathbb{N}_+}$, $\{b^k\}_{k\in\mathbb{N}_+}$ *and* $\{c^k\}_{k\in\mathbb{N}_+}$ *be three non-negative scalar sequences such that* $\sum_{k=0}^{\infty}c^k < \infty$ *and*

$$a^{k+1} \leq a^k - b^k + c^k, \tag{1.13}$$

then, the sequence $\{a^k\}_{k\in\mathbb{N}_+}$ *must convergence to some constant* $\delta \geq 0$*, i.e.,* $\lim_{k\to\infty}a^k = \delta$*, and the sequence* $\{b^k\}_{k\in\mathbb{N}_+}$ *must be summable, i.e.,* $\sum_{k=0}^{\infty}b^k < \infty$*.*

2. IMPROVING THE CONVERGENCE OF DISTRIBUTED GRADIENT DESCENT VIA INEXACT AVERAGE CONSENSUS

A novel DGDx framework [87] is presented in this chapter, which improves the inexact convergence of the standard DGD algorithm. Starting from a new perspective to understand the cause of inexact convergence, i.e., inaccuracy of consensus results, we incorporate multiple communication iterations into the consensus steps. Instead of explicitly specifying the number of iterations according to the time-step k, we introduce a consensus error bound to directly control the consensus accuracy at each time-step. It is proved that an exact convergence can be achieved by the proposed algorithm when the sequence of consensus error bounds decays to zero; furthermore, it converges at a linear rate when the sequence decays to zero linearly. Due to the flexibility of the choice of both consensus error bound sequences and consensus schemes, the proposed framework offers potential opportunities to balance the communication and computation in distributed optimization.

2.1 Problem Statement – Inexact Convergence of the DGD Algorithm

As mentioned in the previous chapter, the well-accepted understanding of the inexact convergence of the DGD algorithm is the lack of global gradient information, and it goes as follows. It should be noted that the following analysis first appears in [37] and here we provide a brief review for the sake of completeness. Recall that the standard DGD algorithm with a non-zero constant step-size $\alpha^k = \alpha$ carries out the iteration,

$$\mathbf{x}^{k+1} = \mathbf{W}\mathbf{x}^k - \alpha\nabla\mathbf{f}(\mathbf{x}^k). \tag{2.1}$$

Let us now assume that the iteration (2.1) has already converged and denote $\mathbf{x}^\infty$ as the limit of the generated sequence $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$. Taking the limit over k on both sides of (2.1) yields $\mathbf{x}^\infty = \mathbf{W}\mathbf{x}^\infty - \alpha\nabla\mathbf{f}(\mathbf{x}^\infty)$. Moreover, suppose that a consensus result will be finally obtained at $\mathbf{x}^\infty$, it immediately implies that $\mathbf{x}^\infty = \mathbf{W}\mathbf{x}^\infty$ since the weight matrix $\mathbf{W}$ is assumed to be doubly-stochastic; see Assumption 1.4.12. Given that α is the nonzero constant, it yields the

optimality condition $\nabla \mathbf{f}(\mathbf{x}^\infty) = 0$. Recall that $\nabla \mathbf{f}$ is the aggregate of local gradients ∇f_i's; see definition (1.7), and each agent can only access its own ∇f_i, the optimality condition requires $\nabla f_i(\mathbf{x}_i^\infty) = 0$ to be satisfied for every agent i. This implies the existence of a common minimizer among all $f_i(\cdot)$'s, which does not hold in a general setting of Problem (P). Therefore, the DGD algorithm with constant step-sizes cannot have the exact convergence.

Based on such understanding, many algorithms have been proposed by focusing on the adjustment of local gradients; see a detailed literature review in Section 1.2 in Chapter 1. However, a natural question here is whether the inexact convergence is necessarily due to the lack of global gradient, and furthermore whether the convergence of the DGD algorithm can be improved from other understanding. Next, we explicitly show another perspective to understand the inexact convergence, i.e., the inaccuracy of consensus results.

We shall remark that the general goal for solving Problem (P) in the distributed manner can be separated into two parts: i) to obtain a consensual solution, i.e., $\mathbf{x}_i^k = \mathbf{x}^\star,\ \forall i \in \mathcal{I}$; and ii) to let $\mathbf{x}^\star$ minimize the global objective function $F(\cdot)$. Keep this in mind, the standard DGD algorithm (2.1) can be also separated into two steps, corresponding to such two goals,

$$\text{Gradient Step:} \quad \mathbf{x}^{k+1/2} = \mathbf{x}^k - \alpha \nabla \mathbf{f}(\mathbf{x}^k); \tag{2.2a}$$

$$\text{Consensus Step:} \quad \mathbf{x}^{k+1} = \mathbf{W}\mathbf{x}^{k+1/2}. \tag{2.2b}$$

The gradient step (2.2a) is adopted to achieve the desired minimizer, whereas the consensus step (2.2b) aims to meanwhile enforce a consensual solution among agents. It is worth pointing out that the separated steps (2.2a) – (2.2b) are essentially different from the standard DGD algorithm (2.1), as the combination of (2.2a) and (2.2b) is written in the following form (termed as NEAR-DGD in [88]),

$$\mathbf{x}^{k+1/2} = \mathbf{W}\mathbf{x}^{k-1/2} - \alpha \nabla \mathbf{f}(\mathbf{W}\mathbf{x}^{k-1/2}). \tag{2.3}$$

While it is confirmed that the better convergence can be expected for the NEAR-DGD iteration (2.3); see details in [88], the evaluation of gradients $\nabla \mathbf{f}(\mathbf{x}^k)$ in (1.8) does not rely on the consensus result and thus can be performed in parallel with the consensus step.

Considering that the consensual result is expected by the consensus step, but a single iteration as shown in (2.2b) is far from sufficient to give an accurate result. Subsequently, our question becomes whether one can obtain the better convergence by improving the consensus result in (2.2b). Next, we answer this question by assuming an ideal average consensus procedure. Suppose that we have an ideal average consensus operator, denoted as $\mathcal{C}^{\infty}(\cdot) : \mathbb{R}^{Ip} \to \mathbb{R}^{Ip}$ with the dynamic defined as,

$$\mathbf{x}^{\mathrm{c}} = \mathcal{C}^{\infty}(\mathbf{x}), \tag{2.4}$$

where the input $\mathbf{x} = [\mathbf{x}_1^{\top}, \mathbf{x}_2^{\top}, \cdots, \mathbf{x}_I^{\top}]^{\top} \in \mathbb{R}^{Ip}$ and output $\mathbf{x}^{\mathrm{c}} = [(\mathbf{x}_1^{\mathrm{c}})^{\top}, (\mathbf{x}_2^{\mathrm{c}})^{\top}, \cdots, (\mathbf{x}_I^{\mathrm{c}})^{\top}]^{\top} \in \mathbb{R}^{Ip}$ satisfy $\mathbf{x}_i^{\mathrm{c}} := (1/I) \cdot \sum_{i=1}^{I} \mathbf{x}_i$. We replace the step (2.2b) with the defined ideal average consensus operator, and then combining it with step (2.2a) gives,

$$\mathbf{x}^{k+1} = \mathcal{C}^{\infty}\left(\mathbf{x}^k - \alpha \nabla \mathbf{f}(\mathbf{x}^k)\right). \tag{2.5}$$

Let $\mathcal{C}_i^{\infty}(\cdot)$ take the i-th component of the output of $\mathcal{C}^{\infty}(\cdot)$, i.e., $\mathbf{x}_i^{k+1} = \mathcal{C}_i^{\infty}(\mathbf{x}^k - \alpha \nabla \mathbf{f}(\mathbf{x}^k))$. According to the property of ideal average consensus operator, it holds that

$$\mathbf{x}_i^{k+1} = \frac{1}{I} \cdot \sum_{\mathrm{i}=1}^{I} \mathbf{x}_i^k - \frac{\alpha}{I} \cdot \sum_{i=1}^{I} \nabla f_i(\mathbf{x}_i^k). \tag{2.6}$$

Note that the previous $\mathbf{x}_i^k$'s are also the outputs of the ideal operator $\mathcal{C}^{\infty}(\cdot)$. Thus, we can have $\mathbf{x}_i^k = \mathbf{x}_j^k,\ \forall i, j \in \mathcal{I}$, and furthermore

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \frac{\alpha}{I} \cdot \sum_{i=1}^{I} \nabla f_i(\mathbf{x}_i^k) = \mathbf{x}_i^k - \alpha \nabla F(\mathbf{x}_i^k). \tag{2.7}$$

It can be seen from (2.7) that adopting an ideal consensus computation reduces the standard DGD algorithm to a centralized gradient descent method for each agent. Given the fact that the centralized gradient descent method achieves the exact convergence with constant step-sizes, it has been shown that the exact convergence can be also obtained by the DGD algorithm with ideal average consensus. Therefore, apart from owing to the lack of global gradient information, the inaccuracy of consensus results caused by (2.2b) can be another

explanation for the inexact convergence of the DGD algorithm. Motivated by such understanding, we are now ready to fix the inexact convergence issue by improving the accuracy of consensus results at step (2.2b).

2.2 The DGDx Algorithm

In this section, we develop the framework of the DGDx algorithm. Based on the new perspective to understand the inexact convergence of the DGD algorithm, a natural idea to cope with the issue is incorporating multiple iterations into the consensus step (2.2b) to improve the accuracy. In fact, a recent paper [88], which agrees with such idea, proposes an algorithm, termed as NEAR-DGD$^+$. Instead of performing just one single iteration as (2.2b), the NEAR-DGD$^+$ algorithm incorporates t iterations into the consensus step at the time-step k, i.e., $\mathbf{x}^{k+1} = (\mathbf{W})^t\mathbf{x}^k$ where $(\mathbf{W})^t$ represent the t-th power of the matrix $\mathbf{W}$. By letting t increase at a rate of $\mathcal{O}(k)$, the NEAR-DGD$^+$ algorithm achieves a linear convergence rate. Here, we will not fix the number of iterations t specifically. On the contrary, at each time-step k, we adopt a consensus error bound $\varepsilon^k \in \mathbb{R}_+$ to guarantee the sufficiently accurate consensus result. Furthermore, considering that the ultimate goal of a consensus step is to ensure an accurate output, we do not specify our consensus scheme either. Instead, we extract the consensus step by an abstract average consensus procedure, denoted as an ε-inexact average consensus operator $\mathcal{C}^\varepsilon(\cdot) : \mathbb{R}^{Ip} \to \mathbb{R}^{Ip}$. The dynamic of $\mathcal{C}^\varepsilon(\cdot)$ follows $\mathbf{x}^{\text{in}} = \mathcal{C}^\varepsilon(\mathbf{x})$, where $\mathbf{x}^{\text{in}} = [(\mathbf{x}_1^{\text{in}})^\top, (\mathbf{x}_2^{\text{in}})^\top, \cdots, (\mathbf{x}_I^{\text{in}})^\top]^\top \in \mathbb{R}^{Ip}$ has $\mathbf{x}_i^{\text{in}} = \mathcal{C}_i^\varepsilon(\mathbf{x})$, and ε represents the consensus error bound satisfied with

$$\left\|\mathbf{x}_i^{\text{in}} - \frac{1}{I}\cdot\sum_{i=1}^{I}\mathbf{x}_i\right\| \leq \varepsilon, \quad \forall i \in \mathcal{I}. \tag{2.8}$$

With the ε-inexact average consensus defined as above, we are now in the position to give our framework of the DGDx algorithm. It performs the following update at each time-step k, with a given consensus error bound ε^k,

$$\mathbf{x}^{k+1} = \mathcal{C}^{\varepsilon^k}\left(\mathbf{x}^k - \alpha\nabla\mathbf{f}(\mathbf{x}^k)\right). \tag{2.9}$$

The detailed steps of the DGDx framework are outlined as Algorithm 1.

Algorithm 1: DGDx Algorithm

Data: Given a constant step-size α and a sequence of consensus error bounds $\{\varepsilon^k\}_{k\in\mathbb{N}_+}$, each agent i initializes its state $\mathbf{x}_i^0$. Let $k=0$.

while a termination criterion is NOT satisfied **do**

Each agent $i \in \mathcal{I}$ **simultaneously** does

`(S.1)` Perform a local gradient step,

$$\mathbf{x}_i^{k+1/2} = \mathbf{x}_i^k - \alpha \nabla f_i(\mathbf{x}_i^k); \tag{2.10}$$

`(S.2)` Carry out the ε-inexact average consensus operator $\mathcal{C}^{\varepsilon^k}(\cdot)$ with error bound ε^k,

$$\mathbf{x}_i^{k+1} = \mathcal{C}_i^{\varepsilon^k}(\mathbf{x}^{k+1/2}); \tag{2.11}$$

`(S.3)` Let $k \leftarrow k+1$ and continue.

end

Before proceeding to convergence analysis, it is worthy making a few remarks on the DGDx algorithm. First, for a given consensus scheme, the slower decaying sequence $\{\varepsilon^k\}_{k\in\mathbb{N}_+}$ may result in more gradient steps to reach the desired optimality, but fewer communication iterations are required for one consensus step. This offers the potential to balance the communication and computation in the implementation of distributed algorithm. Second, it is emphasized that we do not specify any consensus scheme inside the black-box operator $\mathcal{C}^{\varepsilon}(\cdot)$. The flexibility of choices of consensus schemes provides the opportunity to further accelerate the convergence of the DGDx algorithm. For instance, if a finite-time convergent consensus scheme can be applied inside $\mathcal{C}^{\varepsilon}(\cdot)$, it is expected that the convergence rate of distributed algorithm will be as fast as the centralized gradient descent method, as shown in Section 2.1. Moreover, such a flexible framework has reduced the distributed optimization problem into an average consensus problem; one only needs to focus on the implementation of $\mathcal{C}^{\varepsilon}(\cdot)$. In this sense, it provides underlying guidelines for further developments on distributed algorithms, such as those considering communication issues (quantization/delay), cyber security issues (link failure/node attack), and asynchrony issues. In particular, we introduce a new average consensus scheme in the following Section 2.4, which only requires the row-stochastic weight

matrix $\mathbf{W}$ in contrast to the doubly-stochastic one as demanded by the standard weighted averaging scheme; see the iteration (2.2b). Third, we also remark that the convergence result of the DGDx algorithm; see Theorem 2.3.1 in the next section, is consistent with the one of NEAR-DGD$^+$ algorithm presented in [88]. In fact, the NEAR-DGD$^+$ algorithm uses weighted averaging scheme and increases the number of iterations as $\mathcal{O}(k)$ in the consensus step, in order to achieve linearly decaying consensus errors. Therefore, the linear convergence rate of the NEAR-DGD$^+$ algorithm can be viewed as an instance of convergence result of our DGDx algorithm; see statement 2) in Theorem 2.3.1. In addition, we argue that the theoretical results obtained here are more general than the one presented in [88], given the following reasons: i) the case with a generally decaying error bound sequence (not necessarily linearly) is discussed; ii) the balance of computation and communication can be realized by the choice of both error bound sequences and consensus schemes; and iii) the same convergence rate with less communication cost can be expected by faster consensus schemes, such as the ones which can achieve finite-time convergence [89], [90].

2.3 Convergence Analysis

We begin the convergence analysis by introducing the following additional notations. Recall that $\mathbf{x}^{k+1/2}$ represents the intermediate result obtained by a pure gradient step, as shown in (2.2a), we denote the exact average of $\mathbf{x}^k$ and $\mathbf{x}^{k+1/2}$ as $\bar{\mathbf{x}}^k \in \mathbb{R}^p$ and $\bar{\mathbf{x}}^{k+1/2} \in \mathbb{R}^p$, respectively, i.e.,

$$\bar{\mathbf{x}}^k := \frac{1}{I} \cdot \sum_{i=1}^{I} \mathbf{x}_i^k \quad \text{and} \quad \bar{\mathbf{x}}^{k+1/2} := \frac{1}{I} \cdot \sum_{i=1}^{I} \mathbf{x}_i^{k+1/2}. \tag{2.12}$$

Moreover, we use $\bar{\mathbf{g}}^k \in \mathbb{R}^p$ to denote the global gradient at the $\bar{\mathbf{x}}^k$, and use $\mathbf{g}^k \in \mathbb{R}^p$ to denote the average of local gradient at $\mathbf{x}^k$, i.e.,

$$\bar{\mathbf{g}}^k := \frac{1}{I} \cdot \sum_{i=1}^{I} \nabla f_i(\bar{\mathbf{x}}^k) \quad \text{and} \quad \mathbf{g}^k := \frac{1}{I} \cdot \sum_{i=1}^{I} \nabla f_i(\mathbf{x}_i^k). \tag{2.13}$$

Next, we show by the following Lemma 2.3.1 that, once the ε-inexact average consensus is guaranteed by $\mathcal{C}^{\varepsilon^k}(\cdot)$, the distance $\|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1}\|$ can be bounded with the error ε^k.

Lemma 2.3.1. *Given that* $\mathbf{x}^{k+1}$ *is the output of* $\mathcal{C}^{\varepsilon^k}(\mathbf{x}^{k+1/2})$ *and thus* $\|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1/2}\| \leq \varepsilon^k$ *is guaranteed for* $\forall i \in \mathcal{I}$*, then one can have,*

$$\|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1}\| \leq 2\varepsilon^k, \quad \forall i \in \mathcal{I}. \tag{2.14}$$

Proof. By the definition of $\bar{\mathbf{x}}^{k+1}$, it holds that

$$\begin{aligned} \|\bar{\mathbf{x}}^{k+1} - \bar{\mathbf{x}}^{k+1/2}\| &= \Big\|\frac{1}{I} \cdot \Big(\sum_{i=1}^{I} \mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1/2}\Big)\Big\| \\ &\leq \frac{1}{I} \cdot \sum_{i=1}^{I} \|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1/2}\| \leq \varepsilon^k. \end{aligned} \tag{2.15}$$

Therefore, for $\forall i \in \mathcal{I}$, one can have

$$\|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1}\| \leq \|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1/2}\| + \|\bar{\mathbf{x}}^{k+1/2} - \bar{\mathbf{x}}^{k+1}\| \leq 2\varepsilon^k, \tag{2.16}$$

□

With the help of Lemma 2.3.1, we next bound the distance between $\bar{\mathbf{x}}^k$ and the optimizer $\mathbf{x}^\star$. Before proceeding to that, we will need a few supporting lemmas.

Lemma 2.3.2. *Suppose that Assumption 1.4.3 holds and let* $L := (1/I) \cdot \sum_{i=1}^{I} L_i^f$*, then one can have*

$$\|\mathbf{g}^k - \bar{\mathbf{g}}^k\| \leq 2L\varepsilon^{k-1}. \tag{2.17}$$

Proof. By the definitions of $\mathbf{g}^k$ and $\bar{\mathbf{g}}^k$, as shown in (2.13), it holds that

$$\begin{aligned} \|\mathbf{g}^k - \bar{\mathbf{g}}^k\| &\leq (1/I) \cdot \sum_{i=1}^{I} \|\nabla f_i(\mathbf{x}_i^k) - \nabla f_i(\bar{\mathbf{x}}^k)\| \\ &\overset{(2.1.a)}{\leq} (1/I) \cdot \sum_{i=1}^{I} L_i^f \|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\| \\ &\overset{(2.1.b)}{\leq} 2L\varepsilon^{k-1}, \end{aligned} \tag{2.18}$$

where (2.1.a) is due to Assumption 1.4.3 and (2.1.b) follows from Lemma 2.3.1. □

Lemma 2.3.3. *Suppose that Assumption 1.4.3 and 1.4.5 hold, then for* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$*, then one can have*

$$\Big(\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\Big)^\top (\mathbf{x} - \mathbf{y}) \geq \frac{1}{\mu + L}\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\|^2 + \frac{\mu L}{\mu + L}\|\mathbf{x} - \mathbf{y}\|^2, \tag{2.19}$$

where L *is defined in Lemma 2.3.2.*

Proof. The proof can be found in Theorem 2.1.11 in [91], and thus omitted. □

Now, we bound the distance $\|\bar{\mathbf{x}}^{k+1} - \mathbf{x}^\star\|$ by the following lemma.

Lemma 2.3.4. *Suppose that Assumptions 1.4.3 and 1.4.5 hold, let the constant step-size satisfy* $\alpha \leq 2/(\mu + L)$ *and the sequence* $\{\varepsilon^k\}_{k\in\mathbb{N}_+}$ *satisfy* $\varepsilon^{k-1} \geq \varepsilon^k$*, then one can have*

$$\|\bar{\mathbf{x}}^{k+1} - \mathbf{x}^\star\| \leq c_1\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\| + c_2\varepsilon^{k-1}, \tag{2.20}$$

where $c_1 = \sqrt{1 - 2\alpha\mu L/(\mu + L)} < 1$ *and* $c_2 = 2L\alpha + 1$.

Proof. Notice that

$$\begin{aligned}
\|\bar{\mathbf{x}}^k - \mathbf{x}^\star - \alpha\bar{\mathbf{g}}^k\|^2 &= \|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2 + \alpha^2\|\bar{\mathbf{g}}^k\|^2 - 2\alpha(\bar{\mathbf{x}}^k - \mathbf{x}^\star)^\top\bar{\mathbf{g}}^k \\
&\overset{(2.2.a)}{\leq} \|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2 + \alpha^2\|\bar{\mathbf{g}}^k\|^2 - 2\alpha\Big(\frac{1}{\mu + L}\|\bar{\mathbf{g}}^k\|^2 + \frac{\mu L}{\mu + L}\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2\Big) \\
&= (1 - \frac{2\alpha\mu L}{\mu + L})\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2 + \alpha(\alpha - \frac{2}{\mu + L})\|\bar{\mathbf{g}}^k\|^2 \\
&\overset{(2.2.b)}{\leq} c_1^2\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2,
\end{aligned} \tag{2.21}$$

where $(2.2.a)$ follows from Lemma 2.3.3 and $(2.2.b)$ is due to the assumption $\alpha \leq 2/(\mu + L)$. Therefore, it holds that

$$\begin{aligned}
\|\bar{\mathbf{x}}^{k+1} - \mathbf{x}^\star\| &\overset{(2.3.a)}{\leq} \|\bar{\mathbf{x}}^{k+1} - \bar{\mathbf{x}}^{k+1/2}\| + \|\bar{\mathbf{x}}^k - \mathbf{x}^\star - \alpha\mathbf{g}^k\| \\
&\overset{(2.3.b)}{\leq} \varepsilon^k + \|\bar{\mathbf{x}}^k - \mathbf{x}^\star - \alpha\bar{\mathbf{g}}^k\| + \alpha\|\mathbf{g}^k - \bar{\mathbf{g}}^k\| \\
&\overset{(2.3.c)}{\leq} \varepsilon^k + c_1\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\| + 2L\alpha\varepsilon^{k-1} \\
&\overset{(2.3.d)}{\leq} c_2\varepsilon^{k-1} + c_1\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|,
\end{aligned} \tag{2.22}$$

where (2.3.a) is due to the fact that $\bar{\mathbf{x}}^{k+1/2} = \bar{\mathbf{x}}^k - \alpha \mathbf{g}^k$, (2.3.$b$) follows from Lemma 2.3.1, (2.3.c) is due to (2.21) and Lemma 2.3.2, and (2.3.d) is due to the assumption $\varepsilon^{k-1} \geq \varepsilon^k$. □

With the help of all the above lemmas, we are now ready to state the main theorem on the convergence of our DGDx algorithm.

Theorem 2.3.1. *Suppose that Assumptions 1.4.3 and 1.4.5 hold, let the step-size satisfy $\alpha < 2/(\mu + L)$, the sequence $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$ generated by Algorithm 1 has:*

i) if the sequence $\{\varepsilon^k\}_{k\in\mathbb{N}_+}$ has $\lim_{k\to\infty} \varepsilon^k = 0$ and $\varepsilon^{k-1} \geq \varepsilon^k$, then one can have the convergence $\lim_{k\to\infty} \|\mathbf{x}_i^k - \mathbf{x}^\star\| = 0$, $\forall i \in \mathcal{I}$;

ii) if, in particular, the sequence $\{\varepsilon^k\}_{k\in\mathbb{N}_+}$ decays to zero linearly, i.e., there exist $c_{\mathrm{e}} > 0$ and $0 < \rho_{\mathrm{e}} < 1$ such that $\varepsilon^k \leq c_{\mathrm{e}}(\rho_{\mathrm{e}})^k$, then there exist $c > 0$ and $0 < \rho < 1$ such that $\|\mathbf{x}_i^k - \mathbf{x}^\star\| \leq c(\rho)^k$, $\forall i \in \mathcal{I}$.

Proof. By the result obtained from Lemma 2.3.4, it holds that

$$\begin{aligned}\|\bar{\mathbf{x}}^{k+1} - \mathbf{x}^\star\| &\leq c_2\varepsilon^{k-1} + c_1\|\bar{\mathbf{x}}^k - \mathbf{x}^\star\| \\ &\leq c_2\varepsilon^{k-1} + c_1\Big(c_2\varepsilon^{k-2} + c_1\|\bar{\mathbf{x}}^{k-1} - \mathbf{x}^\star\|\Big) \\ &\leq (c_1)^k\|\bar{\mathbf{x}}^1 - \mathbf{x}^\star\| + c_2\sum_{t=0}^{k-1}(c_1)^{k-1-t}\varepsilon^t.\end{aligned} \tag{2.23}$$

According to Lemma 2.3.4, we can have that $c_1 < 1$ and thus

$$\begin{aligned}\lim_{k\to\infty}\|\mathbf{x}_i^{k+1} - \mathbf{x}^\star\| &\leq \lim_{k\to\infty}\|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}^{k+1}\| + \lim_{k\to\infty}\|\bar{\mathbf{x}}^{k+1} - \mathbf{x}^\star\| \\ &\overset{(2.4.a)}{\leq} \lim_{k\to\infty} 2\varepsilon^k + \lim_{k\to\infty}(c_1)^k\|\bar{\mathbf{x}}^1 - \mathbf{x}^\star\| + \lim_{k\to\infty} c_2\sum_{t=0}^{k-1}(c_1)^{k-1-t}\varepsilon^t \\ &\overset{(2.4.b)}{=} 0,\end{aligned} \tag{2.24}$$

where (2.4.a) is due to (2.23) and Lemma 2.3.1, and (2.4.b) follows from the fact $c_1 < 1$ and Lemma 1.4.2 in Chapter 1. Therefore, the statement i) in Theorem 2.3.1 is proved.

Now, we consider statement ii) where the sequence of consensus error bounds decays linearly. Following the same path that we proved the statement i), let us first show that the

distance $\|\bar{\mathbf{x}}^{k+1}-\mathbf{x}^\star\|$ converges to zero at a linear rate, i.e., there exist constants $c'>0$ and $0<\rho'=\max\{\rho_{\mathrm{e}},c_1+c_2c_{\mathrm{e}}/(c'\rho_{\mathrm{e}})\}<1$ such that

$$\|\bar{\mathbf{x}}^{k+1}-\mathbf{x}^\star\|\leq c'(\rho')^{k+1}. \tag{2.25}$$

Recall the inequality (2.20) obtained by Lemma 2.3.4, we prove the statement by induction. First, (2.25) apparently holds when $k=0$. Suppose that (2.25) holds at the k-th iteration, we now consider the $(k+1)$-th iteration and have,

$$\begin{aligned}\|\bar{\mathbf{x}}^{k+1}-\mathbf{x}^\star\|&\leq c_2\varepsilon^{k-1}+c_1\|\bar{\mathbf{x}}^{k}-\mathbf{x}^\star\|\\&\leq c_2c_{\mathrm{e}}\cdot(\rho_{\mathrm{e}})^{k-1}+c_1c'\cdot(\rho')^k\\&=c'(\rho')^k\Big(c_1+\frac{c_2c_{\mathrm{e}}}{c'\rho_{\mathrm{e}}}(\frac{\rho_{\mathrm{e}}}{\rho'})^k\Big)\\&\overset{(2.5.a)}{\leq} c'(\rho')^{k+1},\end{aligned} \tag{2.26}$$

where $(2.5.a)$ is due to the fact $\rho'=\max\{\rho_{\mathrm{e}},c_1+c_2c_{\mathrm{e}}/(c'\rho_{\mathrm{e}})\}$. Now, following the same path as shown in (2.24), there exist constants $c=\max\{2c_{\mathrm{e}}/\rho_{\mathrm{e}},c'\}$ and $\rho=\max\{\rho_{\mathrm{e}},\rho'\}$ such that

$$\|\mathbf{x}_i^{k+1}-\mathbf{x}^\star\|\leq\|\mathbf{x}_i^{k+1}-\bar{\mathbf{x}}^{k+1}\|+\|\bar{\mathbf{x}}^{k+1}-\mathbf{x}^\star\|\overset{(2.6.a)}{\leq}2c_{\mathrm{e}}\cdot(\rho_{\mathrm{e}})^k+c'\cdot(\rho')^{k+1}\leq c\cdot(\rho)^{k+1}, \tag{2.27}$$

where $(2.6.a)$ follows from Lemma 2.3.1. Therefore, the proof is completed. □

2.4 Average Consensus with Row-Stochastic Weight Matrices

In this section, we study a novel average consensus scheme which only involves the row-stochastic weight matrices, as distinct from the classical weighted averaging scheme (2.2b) which requires the weight matrix to be doubly-stochastic. It should be highlighted that, in general, the doubly-stochastic weight matrix is difficult or even impossible to be obtained in the distributed way, especially when the underlying multi-agent network follows a directed graph. On the contrary, the row-stochastic weight matrix can be simply realized by each agent locally and simultaneously. For instance, assuming that each agent i knows its in-degree $d_{i,+}$ (this knowledge can be readily obtained by counting the number of pieces of

received information), one of the simplest to assign the row-stochastic matrix is to let the corresponding weight be $w_{ij} = 1/d_{i,+}$. Although some push-sum based protocols [92], [93] are proposed to be compliant with the directed graphs, they usually assume the weight matrix to be column-stochastic, and we remark that a scheme which requires only row-stochastic weight matrices is arguably more applicable than the one requiring the column-stochastic ones. A good example is when we consider the underlying multi-agent network is under attack; more details on this issue can be found in Chapter 4.

Recall that a general average consensus scheme expects all agents within the network to identically achieve the average of their initial states, i.e.,

$$\lim_{k\to\infty} \|\mathbf{x}_i^k - \frac{1}{I}\cdot\sum_{i=1}^{I}\mathbf{x}_i^0\| = 0, \quad \forall i \in \mathcal{I}, \tag{2.28}$$

where $\mathbf{x}_i^0$ denotes each agent's initial state. To this end, we introduce two auxiliary variables $\boldsymbol{\phi}_i^k \in \mathbb{R}^I$ and $\boldsymbol{\xi}_i^k \in \mathbb{R}^p$ (with initialization $\boldsymbol{\phi}_i^0 = \mathbf{e}_i$ and $\boldsymbol{\xi}_i^0 = \mathbf{x}_i^0$) for each agent i, and let all agents simultaneously perform the following update of their local variables,

$$\begin{aligned} \boldsymbol{\phi}_i^{k+1} &= \sum_{j\in\mathcal{N}_{i,\text{in}}} w_{ij}\boldsymbol{\phi}_j^k; \\ \boldsymbol{\xi}_i^{k+1} &= \sum_{j\in\mathcal{N}_{i,\text{in}}} w_{ij}\boldsymbol{\xi}_j^k + \left(\frac{1}{\mathcal{I}_i\left(\boldsymbol{\phi}_i^{k+1}\right)} - \frac{1}{\mathcal{I}_i\left(\boldsymbol{\phi}_i^k\right)}\right)\cdot\mathbf{x}_i^0. \end{aligned} \tag{2.29}$$

and output the states $\mathbf{x}_i^{k+1} = (1/I)\cdot\boldsymbol{\xi}_i^{k+1}$ at each iteration k. Note that, in (2.29), the operator $\mathcal{I}_i(\cdot) : \mathbb{R}^I \to \mathbb{R}$ selects the i-th component of the input vector. Let us emphasize again that we here consider the underlying multi-agent network to follow the strongly-connected directed graph; see Assumption 1.4.7 and assume the compliant weight matrix $W = [w_{ij}]_{i,j=1}^I$ to be row-stochastic; see Assumptions 1.4.9 and 1.4.10.

To perform the convergence analysis of our new average consensus scheme, let us rewrite the iteration (2.29) into the compact form, as we did for the DGD algorithm. Note that, due to the specific initialization $\boldsymbol{\phi}_i^0 = \mathbf{e}_i$, each $\boldsymbol{\phi}_i^k$ is essentially the i-th column of the matrix $(W)^k$.

Consequently, the update of the aggregated variable $\boldsymbol{\xi}^k = [(\boldsymbol{\xi}_1^k)^\top, (\boldsymbol{\xi}_2^k)^\top, \cdots, (\boldsymbol{\xi}_I^k)^\top]^\top \in \mathbb{R}^{Ip}$ is equivalent to

$$\boldsymbol{\xi}^{k+1} = \mathbf{W}\boldsymbol{\xi}^k + \left(\left(\mathcal{D}^{-1}_{(W)^{k+1}} - \mathcal{D}^{-1}_{(W)^k}\right) \otimes \mathbf{I}_p\right) \cdot \mathbf{x}^0, \tag{2.30}$$

and thus $\mathbf{x}^{k+1} = (1/I) \cdot \boldsymbol{\xi}^{k+1}$, where the matrix $\mathcal{D}^{-1}_{(W)^k} \in \mathbb{R}^{I\times I}$ first takes the diagonal entries of the $(W)^k$ and then generates the inverse of the obtained diagonal matrix, i.e., $\mathcal{D}^{-1}_{(W)^k} = \text{diag}\{1/w_{11}, 1/w_{22}, \cdots, 1/w_{II}\}$.

With the above compact form of the average consensus scheme, we now provide its convergence result as the following theorem.

Theorem 2.4.1. *Under Assumptions 1.4.7, 1.4.9 and 1.4.10, the sequence $\{\mathbf{x}^k\}_{k\in\mathbb{N}^+}$ generated by iteration (2.30) or equivalently (2.29) has the convergence*

$$\lim_{k\to\infty} \mathbf{x}_i^k = \frac{1}{I} \cdot (\mathbf{1}_I^\top \otimes \mathbf{I}_p) \cdot \mathbf{x}^0, \ \forall i \in \mathcal{I}. \tag{2.31}$$

Proof. By the iteration (2.30), we can have

$$\begin{aligned} \boldsymbol{\xi}^{k+1} &= (\mathbf{W})^{k+1}\boldsymbol{\xi}^0 + \sum_{t=0}^{k} (\mathbf{W})^{k-t}\left(\left(\mathcal{D}^{-1}_{(W)^{t+1}} - \mathcal{D}^{-1}_{(W)^t}\right) \otimes \mathbf{I}_p\right) \cdot \mathbf{x}^0 \\ &= \left((W)^{k+1} + \sum_{t=0}^{k} (W)^{k-t}\left(\mathcal{D}^{-1}_{(W)^{t+1}} - \mathcal{D}^{-1}_{(W)^t}\right)\right) \otimes \mathbf{I}_p \cdot \mathbf{x}^0. \end{aligned} \tag{2.32}$$

Note that the last equality is due to the fact $\mathbf{W} = W \otimes \mathbf{I}_p$ and the initialization $\boldsymbol{\xi}^0 = \mathbf{x}^0$. Next, to facilitate the proof, let us define a new matrix $P^{k+1} \in \mathbb{R}^{I\times I}$ as

$$P^{k+1} = (W)^{k+1} + \sum_{t=0}^{k} (W)^{k-t}\left(\mathcal{D}^{-1}_{(W)^{t+1}} - \mathcal{D}^{-1}_{(W)^t}\right). \tag{2.33}$$

Then, the dynamics of the state $\mathbf{x}^{k+1}$ can be simplified as $\mathbf{x}^{k+1} = (1/I) \cdot (P^{k+1} \otimes \mathbf{I}_p) \cdot \mathbf{x}^0$. Now, in order to prove the theorem, it suffices to show the following convergence

$$\lim_{k\to\infty} \|P^k - \mathbf{1}_I\mathbf{1}_I^\top\| = 0. \tag{2.34}$$

According to Lemma 1.4.1 in Chapter 1, it has been already known that, under Assumptions 1.4.7, 1.4.9 and 1.4.10, we can have $\lim_{k\to\infty}\|(W)^k - \mathbf{1}_I\boldsymbol{\pi}^\top\| = 0$. Next, let us suppose that there exists an index $\bar{k} > 0$ such that the difference $\mathcal{D}^{-1}_{(W)^{k+1}} - \mathcal{D}^{-1}_{(W)^k}$ is small enough for all $k > \bar{k}$, and we denote,

$$P^{k,\bar{k}} = (W)^{k+1} + \sum_{t=0}^{\bar{k}}(W)^{k-t}\Big(\mathcal{D}^{-1}_{(W)^{t+1}} - \mathcal{D}^{-1}_{(W)^t}\Big). \tag{2.35}$$

Applying the above Lemma 1.4.1 again, we can have that $\lim_{k\to\infty} P^{k,\bar{k}}$ exists and it only depends on the index $\bar{k}$. Thus, let us denote the limit as another matrix $Q^{\bar{k}} \in \mathbb{R}^{I\times I}$ where

$$Q^{\bar{k}} = \lim_{k\to\infty} P^{k,\bar{k}} = \mathbf{1}_I\boldsymbol{\pi}^\top + \mathbf{1}_I\boldsymbol{\pi}^\top \cdot \Big(\mathcal{D}^{-1}_{(W)^{\bar{k}+1}} - \mathcal{D}^{-1}_{(W)^0}\Big). \tag{2.36}$$

Now, we re-change the index of the sequence $\{Q^{\bar{k}}\}_{k\in\mathbb{N}_+}$ from $\bar{k}$ to k and further have,

$$\lim_{k\to\infty} Q^k = \mathbf{1}_I\boldsymbol{\pi}^\top \cdot \lim_{k\to\infty} \mathcal{D}^{-1}_{(W)^{k+1}} = \mathbf{1}_I\mathbf{1}_I^\top. \tag{2.37}$$

Therefore, it holds that

$$\begin{aligned}
&\|P^k - \mathbf{1}_I\mathbf{1}_I^\top\| \leq \|P^k - Q^k\| + \|Q^k - \mathbf{1}_I\mathbf{1}_I^\top\| \\
&\leq \|(W)^{k+1} - \mathbf{1}_I\boldsymbol{\pi}^\top\| + \Big\|\sum_{t=0}^{k}\Big((W)^{k-t} - \mathbf{1}_I\boldsymbol{\pi}^\top\Big)\Big(\mathcal{D}^{-1}_{(W)^{t+1}} - \mathcal{D}^{-1}_{(W)^t}\Big)\Big\| + \|Q^k - \mathbf{1}_I\mathbf{1}_I^\top\| \\
&\leq \|(W)^{k+1} - \mathbf{1}_I\boldsymbol{\pi}^\top\| + \sum_{t=0}^{k} c_0\rho_0^{k-t}\|\mathcal{D}^{-1}_{(W)^{t+1}} - \mathcal{D}^{-1}_{(W)^t}\| + \|Q^k - \mathbf{1}_I\mathbf{1}_I^T\|.
\end{aligned} \tag{2.38}$$

Note that the first two inequalities are due to the definitions of matrices P^k and Q^k, as well as the triangle inequality; the last one is by Lemma 1.4.1 in Chapter 1. Now, based on the facts that $\lim_{k\to\infty}\|(W)^k - \mathbf{1}_I\boldsymbol{\pi}^\top\| = 0$ and $\lim_{k\to\infty}\|(Q)^k - \mathbf{1}_I\mathbf{1}_I^\top\| = 0$, as well as Lemma 1.4.2 in Chapter 1, it is straightforward to verify that $\lim_{k\to\infty}\|(P)^k - \mathbf{1}_I\boldsymbol{\pi}^\top\| = 0$. Therefore, the proof is complete. □

2.5 ε-Inexact Average Consensus

Although our framework of the DGDx algorithm is well-developed and its convergence has also been proved by Theorem 2.3.1, yet the implementation of the ε-inexact average consensus operator $\mathcal{C}^{\varepsilon}(\cdot)$ remains unclear especially in the distributed environment. The main challenge here is the lack of global awareness of the timing when all agents reach the desired consensus error bound; see the definition in (2.8). In this section, we aim to fix this issue and inspired by the protocol in [94], we leverage the idea of maximum and minimum consensus to design our own scheme.

Since the principles behind both maximum and minimum consensus schemes follow the same path, here we only introduce the maximum consensus scheme and it can be easily extended into the minimum consensus with slight changes. A maximum consensus scheme for a scalar system aims at enforcing every agent to reach a consensual state $\bar{z}^{\max} = \max_{i\in\mathcal{I}} \bar{z}_i^0$, where $\bar{z}_i^0$ is the scalar initial state maintained by agent i. For a vector system where the initial states are represented as p-dimensional vectors $\bar{\mathbf{z}}_i^0 = [\bar{z}_i^0(1), \bar{z}_i^0(2), \cdots, \bar{z}_i^0(p)]^\top \in \mathbb{R}^p$, we set the goal as achieving the element-wise maximum state, i.e.,

$$\bar{\mathbf{z}}^{\max} = [\max_{i\in\mathcal{I}} \bar{z}_i^0(1),\ \max_{i\in\mathcal{I}} \bar{z}_i^0(2),\ \cdots,\ \max_{i\in\mathcal{I}} \bar{z}_i^0(p)]^\top \in \mathbb{R}^p. \tag{2.39}$$

Keep this in mind, our maximum consensus scheme for a vector system is

$$\bar{z}_i^{k+1}(t) = \max_{j\in\mathcal{N}_i} \bar{z}_j^k(t), \quad \forall t \in \{1, 2, \cdots, p\},\ i \in \mathcal{I}, \tag{2.40}$$

where $\bar{z}_i^k(t)$ is the t-th element of the p-dimensional vector $\bar{\mathbf{z}}_i^k$ at time-step k. Its convergence is stated as follows.

Proposition 2.5.1. *For any given $\bar{\mathbf{z}}^0 = [(\bar{\mathbf{z}}_1^0)^\top, (\bar{\mathbf{z}}_2^0)^\top, \cdots, (\bar{\mathbf{z}}_I^0)^\top]^\top \in \mathbb{R}^{Ip}$, under Assumption 1.4.7, the sequence $\{\bar{\mathbf{z}}^k\}_{k\in\mathbb{N}_+}$ generated by (2.40) converges to the maximum consensual state in finite D time-steps, where D is the diameter of the time-invariant graph.*

Proof. The proof can be completed by a simple extension of Proposition 2.3 in [94] □

Following the same path, the minimum consensus scheme performs

$$\underline{z}_i^{k+1}(t) = \min_{j\in\mathcal{N}_i} \underline{z}_j^k(t), \quad \forall t \in \{1, 2, \cdots, p\},\ i \in \mathcal{I}, \tag{2.41}$$

and its the convergence result is provided as follows.

Proposition 2.5.2. *For any given* $\underline{\mathbf{z}}^0 = [(\underline{\mathbf{z}}_1^0)^\top, (\underline{\mathbf{z}}_2^0)^\top, \cdots, (\underline{\mathbf{z}}_I^0)^\top]^\top \in \mathbb{R}^{Ip}$*, under Assumption*Assumption 1.4.7*, the sequence* $\{\underline{\mathbf{z}}^k\}_{k\in\mathbb{N}_+}$ *generated by* (2.41) *converges to the minimum consensual state in finite* D *time-steps, which is defined as*

$$\underline{\mathbf{z}}^{\min} = [\min_{i\in\mathcal{I}} \underline{z}_i^0(1),\ \min_{i\in\mathcal{I}} \underline{z}_i^0(2),\ \cdots,\ \min_{i\in\mathcal{I}} \underline{z}_i^0(p)]^\top \in \mathbb{R}^p. \tag{2.42}$$

With the maximum and minimum consensus schemes developed as above, we are now ready to present the ε-inexact average consensus scheme. Considering that both maximum and minimum consensual states can be exactly achieved in the finite D time-steps, the guideline here is to check the stopping criteria for every D steps by measuring the discrepancy between the results obtained by maximum and minimum consensus. We show by the following lemma that the states of agents $\mathbf{y}_i \in \mathbb{R}^p$ must reach the ε-inexact average consensus, if the discrepancy satisfies $\|\bar{\mathbf{y}}^{\max} - \underline{\mathbf{y}}^{\min}\| \leq \varepsilon$.

Lemma 2.5.1. *Given the states* $\mathbf{y}_i = [y_i(1), y_i(2), \cdots, y_i(p)]^\top \in \mathbb{R}^p$ *and outputs of maximum and minimum consensus* $(\bar{\mathbf{y}}^{\max}, \underline{\mathbf{y}}^{\min})$ *whose* t*-th elements has* $\bar{y}^{\max}(t) = \max_{i\in\mathcal{I}} y_i(t)$ *and* $\underline{y}^{\min}(t) = \min_{i\in\mathcal{I}} y_i(t)$*, respectively, if there exists a constant* $\varepsilon > 0$ *such that*

$$\|\bar{\mathbf{y}}^{\max} - \underline{\mathbf{y}}^{\min}\| \leq \varepsilon, \tag{2.43}$$

it implies $\|\mathbf{y}_i - \bar{\mathbf{y}}\| \leq \varepsilon$ *where* $\bar{\mathbf{y}} = (1/I) \cdot \sum_{i=1}^I \mathbf{y}_i$ *denotes the average of* $\mathbf{y}_i$*'s.*

Proof. By the definitions of maximum and minimum consensus, it holds that

$$\underline{y}^{\min}(t) \leq y_i(t) \leq \bar{y}^{\max}(t), \quad \forall i \in \mathcal{I},\ t \in \{1, 2, \cdots, p\}, \tag{2.44}$$

and furthermore,

$$|y_i(t) - y_j(t)| \leq |\bar{y}^{\max}(t) - \underline{y}^{\min}(t)|, \quad \forall i, j \in \mathcal{I},\ t \in \{1, 2, \cdots, p\}. \tag{2.45}$$

Consequently, we have $\|\mathbf{y}_i - \mathbf{y}_j\| \leq \|\bar{\mathbf{y}}^{\max} - \underline{\mathbf{y}}^{\min}\| \leq \varepsilon$, and thus for $\forall i \in \mathcal{I}$,

$$\left\|\mathbf{y}_i - (1/I) \cdot \sum_{j=1}^{I} \mathbf{y}_j\right\| = \left\|(1/I) \cdot \sum_{j=1}^{I} (\mathbf{y}_i - \mathbf{y}_j)\right\| \leq (1/I) \cdot \sum_{j=1}^{I} \|\mathbf{y}_i - \mathbf{y}_j\| \leq \varepsilon. \tag{2.46}$$

□

The above Lemma 2.5.1 guarantees the ε-inexact average consensus (2.8) by ensuring the stopping criteria (2.43). Now, we present the detailed implementation of the operator $\mathcal{C}^{\varepsilon}(\cdot)$, which is outlined as Algorithm 2.

Algorithm 2: ε-Inexact Average Consensus Operator

Data: Given each agent's current state $\mathbf{x}_i$ and the consensus error bound ε, let $\mathbf{y}_i^0 = \mathbf{x}_i$, $\bar{\mathbf{y}}_i^0 = \varepsilon\mathbf{1}$ and $\underline{\mathbf{y}}_i^0 = \mathbf{0}$. Set $k = 0$ and `stop_flag` $= 0$.

while `stop_flag` is NOT true **do**

Each agent $i \in \mathcal{I}$ **simultaneously** does

if $(k \ \texttt{mod}\ D) = 0$ **then**

if $\|\bar{\mathbf{y}}_i^k - \underline{\mathbf{y}}_i^k\| \leq \varepsilon$ **then**

Set `stop_flag` $= 1$;

else

Reinitialize $\bar{\mathbf{y}}_i^k = \mathbf{y}_i^k$ and $\underline{\mathbf{y}}_i^k = \mathbf{y}_i^k$;

end

end

`(S.1)` Perform the one-iteration consensus step, i.e., $\mathbf{y}_i^{k+1} = \mathcal{A}_i^{\text{one}}(\mathbf{y}^k)$;

`(S.2)` Perform the maximum and minimum consensus steps as shown in (2.40);

`(S.3)` Let $k \leftarrow k + 1$ and continue.

end

Note that, in Algorithm 2, we use $\bar{\mathbf{y}}_i^k$ and $\underline{\mathbf{y}}_i^k$ to track the maximum and minimum consensus respectively, and use $\mathbf{y}_i^k$ to obtain the ε-inexact average consensus result. As remarked in Section 2.2, we do not specify the consensus scheme to iterate $\mathbf{y}_i^k$. Instead,

we adopt an abstract one-iteration consensus step $\mathcal{A}^{\text{one}}(\cdot) : \mathbb{R}^{Ip} \to \mathbb{R}^{Ip}$ with $\mathcal{A}_i^{\text{one}}(\cdot)$ being the i-th component of the output, to denote one iteration in a general consensus scheme. Such a step can be implemented by the widely-used weighted averaging as shown in (2.2b), as well as our new average consensus scheme introduced in Section 2.4. Certainly, one can implement it by many other options.

We shall also remark that the stopping criteria checking is D time-steps delayed, compared with the one-iteration consensus step (`S.1`) in Algorithm 2, since the maximum and minimum consensus always take a finite-time period D to reach the desired result. In fact, for many specific consensus schemes such as the weighted averaging, the consensus error can be preserved in each time-step, i.e., if $\|\mathbf{x}^k - \bar{\mathbf{x}}\| \leq \varepsilon$ then $\|\mathbf{x}^{k+1} - \bar{\mathbf{x}}\| \leq \varepsilon$. To show this, let us assume that the weighted averaging runs $\mathbf{x}^{k+1} = \mathbf{W}\mathbf{x}^k$ with aim to achieve the average $\bar{\mathbf{x}}$, and it has reached the ε-inexact average consensus at iteration k, i.e., $\|\mathbf{x}_i^k - \bar{\mathbf{x}}\| \leq \varepsilon,\ \forall i \in \mathcal{I}$. Then, it holds that

$$\|\mathbf{x}_i^{k+1} - \bar{\mathbf{x}}\| = \Big\| \sum_{j=1}^{I} w_{ij}\mathbf{x}_j^k(t) - \bar{\mathbf{x}}(t) \Big\| = \Big\| \sum_{j=1}^{I} w_{ij}\Big(\mathbf{x}_j^k(t) - \bar{\mathbf{x}}(t)\Big) \Big\| \leq \sum_{j=1}^{I} w_{ij}\|\mathbf{x}_j^k(t) - \bar{\mathbf{x}}(t)\| \leq \varepsilon. \tag{2.47}$$

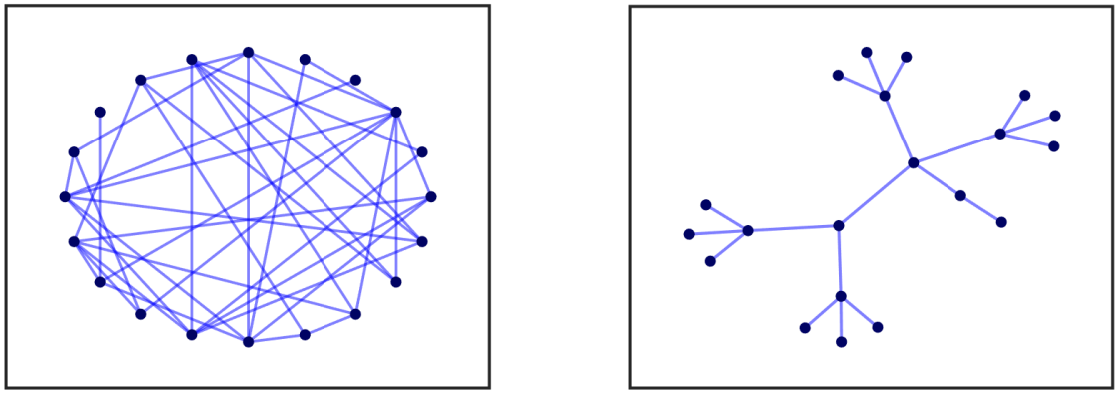
where the second equality follows from the stochasticity of weight matrix $\mathbf{W}$, the first inequality is due to the triangle inequality, and the second inequality follows from the condition that $\|\mathbf{x}_i^k(t) - \bar{\mathbf{x}}(t)\| \leq \varepsilon,\ \forall i \in \mathcal{I}$. Thus, we claim that Algorithm 2 is valid for such consensus schemes. Moreover, if the adopted consensus scheme does not have such a preservation property, an extra buffer will be needed for agents to store the previous states.

2.6 Simulation Result

In this section, we evaluate the proposed DGDx framework on the following distributed Least Squares problem,

$$\underset{\mathbf{x}\in\mathbb{R}^p}{\text{minimize}} \quad F(\mathbf{x}) = \sum_{i=1}^{I} \|\boldsymbol{A}_i\mathbf{x} - \mathbf{b}_i\|^2. \tag{2.48}$$

Here, each agent i is assumed to own a pair of measurements $\{\boldsymbol{A}_i, \mathbf{b}_i\}$. The goal is to minimize the global objective function $F(\mathbf{x})$ in a distributed manner. In the following, we compare four algorithms: the DGDx algorithm with both linearly decaying error bound sequence (termed as L-DGDx) and sub-linearly decaying sequence (termed as Sub-DGDx), the NEAR-DGD$^+$ algorithm, and also the DGD algorithm with diminishing step-sizes (termed as DGD-dim).

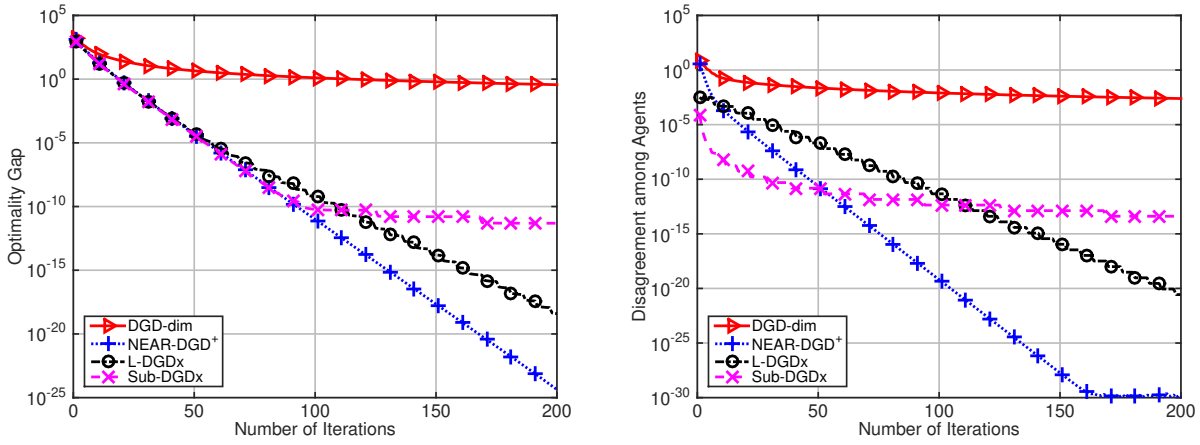


(a) The general undirected graph

(b) The undirected tree graph

Figure 2.1. Topology of the multi-agent network

In this simulation, we consider a network composed of $n = 20$ agents, and assume that each agent takes 20 measurements, i.e., $\mathbf{b}_i \in \mathbb{R}^{20}$. The network topology follows a randomly generated undirected graph, as shown in Fig. 1(a), which is guaranteed to be connected. Note that the algebraic connectivity of the network has $\sigma = 0.686$ and its diameter is $D = 4$. The weight matrix W is generated through the Metropolis-Hasting rule so that W is doubly stochastic. We first evaluate our DGDx algorithm with two different choices of consensus error bound sequence $\{\varepsilon^k\}_{k\in\mathbb{N}_+}$: i) a linearly decaying case, i.e., $\varepsilon^k = (0.9)^k$; and ii) a sub-linearly decaying case, i.e., $\varepsilon^k = 0.1/k$. The weighted averaging scheme is utilized for the consensus steps in the DGDx algorithm. Since the same local descent step is adopted in both NEAR-DGD$^+$ algorithm and our DGDx algorithm, for a fair comparison, we use the same constant step-size $\alpha = 0.4$ in both two algorithms. The diminishing step-sizes $\{\alpha^k\}_{k\in\mathbb{N}_+}$ for standard DGD algorithm is set as $\alpha^{k+1} = \alpha^k(1 - \delta\alpha^k)$ with $\alpha^0 = 0.6$ and $\delta = 0.2$. The performance of algorithms is evaluated as follows.

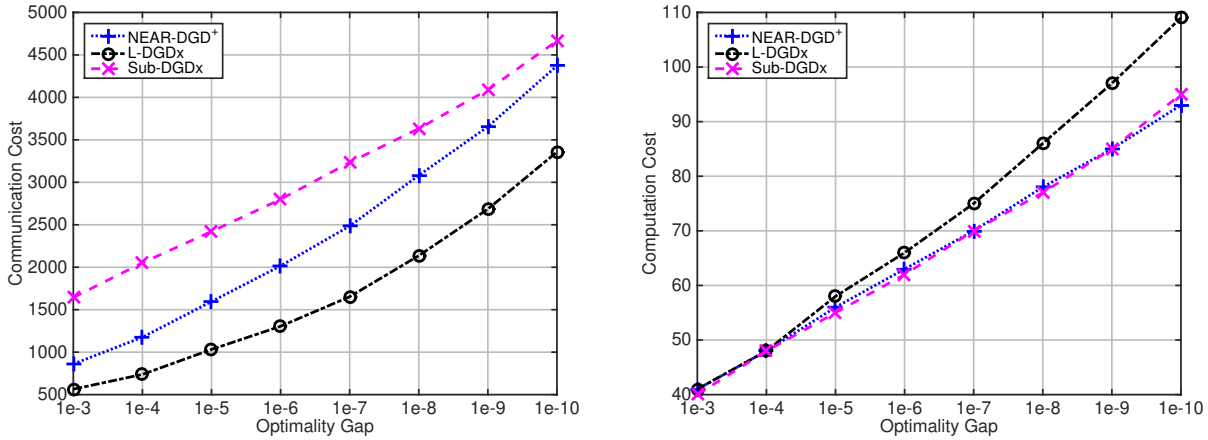


(a) Optimality Gap

(b) Disagreement among agents

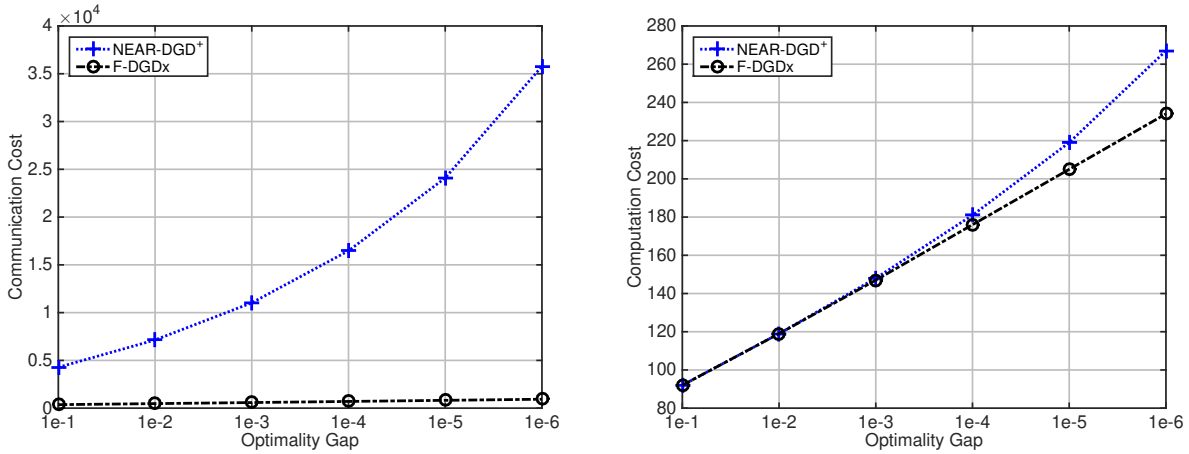
Figure 2.2. Comparison of the convergence results

Fig. 2.2 demonstrates the convergence results of all four algorithms by plotting the optimality gap (see Fig. 2.2(a)) and disagreement among agents (see Fig. 2.2(b)) versus the number of iterations. The optimality gap is measured by $S^k = \|\nabla F(\bar{\mathbf{x}}^k)\|^2$ at each time-step k, where $\bar{\mathbf{x}}^k = (1/I) \cdot \sum_{i=1}^{I} \mathbf{x}_i^k$ represents the average of all agents' states; the disagreement among agents is computed by $D^k = (1/I) \cdot \sum_{i=1}^{I} \|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\|^2$. It can be seen from Fig. 2.2 that our DGDx algorithm (both L-DGDx and Sub-DGDx) achieves the exact convergence, while the L-DGDx shows the faster convergence rate than Sub-DGDx. Although it is observed that the NEAR-DGD$^+$ algorithm can reach the same level of optimality gap by fewer iterations than the DGDx algorithm, we next show that the NEAR-DGD$^+$ algorithm actually consumes more communication resources. The comparison of computation and communication cost is illustrated in Fig. 2.3. While the communication cost at each time-step is counted as the number of iterations involved in the consensus step (see Fig. 2.3(a)), the computation cost is defined as the number of gradient steps (see Fig. 2.3(b)). It is shown that the L-DGDx requires the least amount of communication cost to achieve the same optimality gaps. Therefore, we conclude that our DGDx algorithm offers extra flexibility in the algorithm implementation to reduce the total cost, compared to the NEAR-DGD$^+$ algorithm.



(a) Communication cost (b) Computation cost

Figure 2.3. Comparison of the computation and communication cost



(a) Communication cost (b) Computation cost

Figure 2.4. Comparison of the computation and communication cost (tree graph)

In addition, to further demonstrate the superiority of our DGDx algorithm when applying finite-time consensus schemes, we carry out the following additional simulation where the finite-time consensus scheme presented in [90] is adopted. Given that such finite-time scheme can be only valid when considering a tree graph, here we simulate a network of $n = 20$ agents which has the topology as shown in Fig. 2.1(b). The other simulation settings are the same as before. We also compare the DGDx algorithm with finite-time consensus (termed as F-DGDx) to the NEAR-DGD$^+$ algorithm. For a fair comparison, the same constant step-

size $\alpha = 0.1$ is specified for both algorithms. Here, we only investigate the communication and computation cost of the algorithms. As shown in Fig. 2.4, the F-DGDx algorithm requires much less communication cost than the NEAR-DGD$^+$ algorithm. That is because only $D = 4$ iterations are needed to reach the pure average at each consensus step of our algorithm, while the number of iterations in NEAR-DGD$^+$ grows up exponentially as the time-step k increases.

3. A GENERALIZED NETWORKED PROXIMAL ALGORITHM FOR DISTRIBUTED OPTIMIZATION

This chapter presents a generalized networked proximal algorithm, termed as NetProx, for solving the distributed optimization problem (P). This new algorithm discards the conventional consensus procedure which appears in majority of the existing distributed algorithms, but takes advantage of the classical proximal point method in the context of centralized optimization to ensure the consensual solutions. Observing that the standard DGD algorithm is just a special case of our NetProx framework when the local objective functions are approximated by the first order Taylor expansions, we then generalize the choice of local approximation functions. It is shown that, with the higher order approximation functions, the convergence of the NetProx algorithm will be accelerated to some extent, but meanwhile, it will cost more computational resources for each agent at each iteration. On this account, the flexibility of choices of the approximation functions further helps to balance the computation and communication in distributed optimization.

3.1 Problem Statement – A Revisit to the DGD Algorithm

As we have reviewed in Section 1.2 of Chapter 1, the majority of the existing first-order primal-based distributed algorithms, including the well-studied DGD algorithm, follows the same path – a consensus step together with a gradient step, where the consensus step is designed to enforce the consensual solution among all agents and the gradient step is adopted to meanwhile achieve the optimizer. Now, in order to develop our new algorithmic framework, we present another perspective to study the DGD algorithm by reformulating its iterations.

Let us first recall that the DGD algorithm performs the following update at each agent $i \in \mathcal{I}$ simultaneously and locally,

$$\mathbf{x}_i^{k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} \mathbf{x}_j^k - \alpha^k \nabla f_i(\mathbf{x}_i^k). \tag{3.1}$$

Note that some underlying assumptions are incorporated by the DGD algorithm; see details in Section 1.4.2 in Chapter 1.

We shall highlight that the right-hand side of (3.1) can be viewed as an analytic solution of minimizing the following objective function $\tilde{f}_i(\cdot) : \mathbb{R}^p \to \mathbb{R}$,

$$\tilde{f}_i(\mathbf{x}) := \nabla f_i(\mathbf{x}_i^k)^\top (\mathbf{x} - \mathbf{x}_i^k) + \frac{1}{2\alpha^k} \cdot \sum_{j=1}^{I} w_{ij} \|\mathbf{x} - \mathbf{x}_j^k\|^2. \tag{3.2}$$

Therefore, the iteration of the DGD algorithm can be also expressed as the following form,

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x} \in \mathbb{R}^p} \Big\{ \underbrace{f_i(\mathbf{x}_i^k) + \nabla f_i(\mathbf{x}_i^k)^\top (\mathbf{x} - \mathbf{x}_i^k)}_{\text{linear approximation of } f_i} + \frac{1}{2\alpha^k} \cdot \underbrace{\sum_{j=1}^{I} w_{ij} \|\mathbf{x} - \mathbf{x}_j^k\|^2}_{\text{proximal term}} \Big\}, \ i \in \mathcal{I}. \tag{3.3}$$

We can now interpret the above iteration (3.3) as follows: each agent attempts to minimize the objective function by considering the linear approximation of each $f_i(\cdot)$ and meanwhile tries to stay close to the states of its neighbors by adding the proximal term. Here, the sequence of step-sizes $\{\alpha^k\}_{k \in \mathbb{N}_+}$ is diminishing, meaning that more and more weights will be assigned to the proximal term. As the iteration index k goes to infinite, the consensual solutions must be achieved since $\alpha^k \to 0$. Once we explain the iteration (3.3) from this new perspective, two questions will naturally arise and remain to be answered:

i) Given that the proximal term aims at enforcing the obtained solution to stay close to the points of neighbors, how can one motivate the using of those weights w_{ij}'s? In particular, how can one justify the necessity of double-stochasticity of the weight matrix $W = [w_{ij}]_{i,j=1}^I$ that is typically required by the DGD algorithm?

ii) Aside from the linear approximation of $f_i(\cdot)$, can one adopt some other forms of the approximation function, such as the higher order approximation or even the original function $f_i(\cdot)$ directly?

Here, we first try to answer the question i) by eliminating the weights in (3.3), and thus obtain the following iteration,

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x} \in \mathbb{R}^p} \Big\{ f_i(\mathbf{x}_i^k) + \nabla f_i(\mathbf{x}_i^k)^\top (\mathbf{x} - \mathbf{x}_i^k) + \frac{1}{2\alpha^k} \cdot \sum_{j \in \mathcal{N}_i} \|\mathbf{x} - \mathbf{x}_j^k\|^2 \Big\}, \ i \in \mathcal{I}. \tag{3.4}$$

At the iteration (3.4), each agent basically treats the information received from its neighbors with an equal weight. To see whether such a weight matrix free scheme works or not, we rewrite it back into the DGD-like form and have,

$$\mathbf{x}_i^{k+1} = \frac{1}{d_i} \cdot \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^k - \frac{\alpha^k}{d_i} \cdot \nabla f_i(\mathbf{x}_i^k). \tag{3.5}$$

Note that here d_i denotes the degree of node i, i.e., $d_i = |\mathcal{N}_i|$. It can be observed from (3.5) that each agent i basically assigns the same weight $1/d_i$ to all the received information from its neighbors, and correspondingly, the local gradient is also scaled by $1/d_i$ at each time-step k. In order to proceed our further analysis, let us express the iteration (3.5) into the compact form. Recall that $\mathbf{x}^k \in \mathbb{R}^{Ip}$ and $\nabla \mathbf{f}(\mathbf{x}^k) \in \mathbb{R}^{Ip}$ denote the aggregates of local states $\mathbf{x}_i^k$'s and local gradients $\nabla f_i(\mathbf{x}_i^k)$'s, respectively, then (3.5) is equivalent to

$$\mathbf{x}^{k+1} = \mathbf{M}\mathbf{x}^k + \alpha^k \mathbf{D}^{-1} \nabla \mathbf{f}(\mathbf{x}^k). \tag{3.6}$$

Note that, in (3.6), the hidden weight matrix $\mathbf{M} \in \mathbb{R}^{Ip \times Ip}$ is defined as $\mathbf{M} := (D^{-1}A) \otimes \mathbf{I}_p$ and $\mathbf{D} := D \otimes \mathbf{I}_p \in \mathbb{R}^{Ip \times Ip}$; recall that the degree matrix $D := \mathrm{diag}\{d_1, d_2, \cdots, d_I\} \in \mathbb{R}^{I \times I}$ and the adjacency $A = [a_{ij}]_{i,j=1}^I \in \mathbb{R}^{I \times I}$ have been defined in Chapter 1. It is straightforward to verify that this new weight matrix $\mathbf{M}$ is row-stochastic but not necessarily doubly-stochastic. However, according to the theory of the DGD algorithm, its convergence can only be guaranteed when the weight matrix is doubly-stochastic, see e.g., [29], [32]. We note here that the iteration (3.6) can still drive the generated sequence $\{\mathbf{x}^k\}_{k \in \mathbb{N}_+}$ to the optimal solution under some necessary conditions. This is primarily due to the scaling of the local gradient, i.e., $\mathbf{D}^{-1}\nabla \mathbf{f}(\mathbf{x}^k)$, and we elaborate on it as follows.

Let us first give a brief review on the result presented in [95] which studies the convergence of the standard DGD algorithm (3.1) when the original weight matrix W is only row-stochastic, rather than doubly-stochastic. Denote the vector $\boldsymbol{v} = [v_i]_{i=1}^I \in \mathbb{R}^I$ as the left eigenvector of W corresponding to the eigenvalue 1, i.e., $\boldsymbol{v}^\top W = \boldsymbol{v}^\top$ and $\mathbf{1}^\top \boldsymbol{v} = 1$, and

define the weighted average of local states as $\widehat{\mathbf{x}}^k := \sum_{i=1}^I v_i \mathbf{x}_i^k$. Then, it is straightforward to verify that the dynamics of $\widehat{\mathbf{x}}^k$ can be written as,

$$\widehat{\mathbf{x}}^{k+1} = \widehat{\mathbf{x}}^k - \alpha^k \sum_{i=1}^I v_i \nabla f_i(\mathbf{x}_i^k). \tag{3.7}$$

Consider that the consensus result will be achieved by (1.5) due to the row-stochasticity of the weight matrix W;see the Lemma 1.4.1 in Chapter 1, i.e., $\lim_{k\to\infty} \|\mathbf{x}_i^k - \hat{\mathbf{x}}^k\| = 0$, it can be understood that the multiple agents are cooperatively minimizing a new objective function $\hat{F}(\mathbf{x}) := \sum_{i=1}^I v_i f_i(\mathbf{x})$, instead of the original one $F(\mathbf{x}) = \sum_{i=1}^I f_i(\mathbf{x})$. Indeed, this motivates a new scheme proposed in [96] which introduces an auxiliary variable $\mathbf{y}_i^k \in \mathbb{R}^I$ to rescale the bias of local objective function $v_i f_i(\mathbf{x})$ and performs

$$\mathbf{y}_i^{k+1} = \sum_{j=1}^I w_{ij} \mathbf{y}_j^k, \tag{3.8a}$$

$$\mathbf{x}_i^{k+1} = \sum_{j=1}^I w_{ij} \mathbf{x}_j^k - \alpha^k \frac{\nabla f_i(\mathbf{x}_i^k)}{\mathcal{I}_i(\mathbf{y}_i^k)}, \tag{3.8b}$$

where $\mathcal{I}_i(\cdot) : \mathbb{R}^I \to \mathbb{R}$ takes the i-th element of the input vector. Provided that the variable $\mathbf{y}_i^k$ can track the left eigenvector $\boldsymbol{v}$ by initialized as $\mathbf{y}_i^0 = \mathbf{e}_i$ where $\mathbf{e}_i$ denotes the unit vector, it is easy to see that $\mathcal{I}_i(\mathbf{y}_i^k)$ will converge to the scaling weight v_i and thus (3.8b) is gradually equivalent to the standard DGD algorithm.

Now, let us look back into our derived DGD-like iteration (3.5) or equivalently (3.6), and compare it with (3.8b). Since we assume that the underlying time-invariant network $\mathcal{G}$ is undirected, it can be verified that the vector $\mathbf{d} = [d_i]_{i=1}^I \in \mathbb{R}^I$ is exactly the left eigenvector of the hidden weight matrix $M = D^{-1}A$ corresponding to the eigenvalue 1, i.e., $\mathbf{d}^\top D^{-1} A = \mathbf{d}^\top$. Therefore, instead of adopting the tracking left eigenvector $\mathbf{y}_i^k$ to rescale the bias of local gradient as in (3.8b), the iteration (3.5) actually obtains the left eigenvector $\mathbf{d}$ as a prior knowledge due to the special assignment of M. It can be seen that (3.5) essentially performs the same updates as (3.8a)–(3.8b), and its convergence can be straightforwardly confirmed by applying Theorem 2 in [96].

Based on the analysis presented above, we have inherently answered the first question. That is, once the DGD algorithm is considered from the new perspective as (3.3), the weight matrix can be actually eliminated and a weight matrix free scheme is thus obtained as (3.4). Next, we attempt to approach the second one by developing our new algorithmic framework – the NetProx algorithm.

3.2 The NetProx Algorithm

To answer the second question, we now consider a general case, in which a generic function $g_i(\cdot\,|\,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is adopted to approximate the local objective function $f_i(\cdot)$. To ensure that $g_i(\cdot\,|\,\cdot)$ is a valid approximation of $f_i(\cdot)$, we require it to satisfy the following conditions.

Assumption 3.2.1. *Each of the local approximation functions $g_i(\cdot\,|\,\cdot)$ is assumed to be*

i) $g_i(\cdot\,|\,\mathbf{x})$ is convex on the set $\mathcal{X}$, $\forall \mathbf{x} \in \mathcal{X}$;

ii) the gradient $\nabla g_i(\cdot\,|\,\mathbf{x})$ is L_i^g-Lipschitz continuous on the set $\mathcal{X}$, $\forall \mathbf{x} \in \mathcal{X}$;

iii) $\nabla g_i(\mathbf{x}\,|\,\mathbf{x}) = \nabla f_i(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$.

Note that, from now on, we consider the distributed optimization problem (P) under the compact feasible set $\mathcal{X}$; see Assumption 1.4.2 in Chapter 1. As a result, the gradient ∇f_i on each point $\mathbf{x} \in \mathcal{X}$ must be bounded, i.e., there exists a constant $\kappa > 0$ such that

$$\|\nabla f_i(\mathbf{x})\| \leq \kappa, \quad \forall x \in \mathcal{X},\ \forall i \in \mathcal{I}. \tag{3.9}$$

It should be highlighted that, aside from the previous linear approximation function, i.e., $g_i(\mathbf{x}\,|\,\mathbf{y}) = f_i(\mathbf{y}) + \nabla f_i(\mathbf{y})^\top(\mathbf{x}-\mathbf{y})$, there are also a few other choices of the function $g_i(\cdot\,|\,\cdot)$; for example, the second-order Taylor expansion, i.e.,

$$g_i(\mathbf{x}\,|\,\mathbf{y}) = f_i(\mathbf{y}) + \nabla f_i(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) + \frac{1}{2}\cdot(\mathbf{x}-\mathbf{y})^\top\nabla^2 f_i(\mathbf{y})(\mathbf{x}-\mathbf{y}), \tag{3.10}$$

where $\nabla^2 f_i(\cdot) \in \mathbb{R}^{p\times p}$ denotes the Hessian matrix, and also the original function, i.e., $g_i(\mathbf{x}\,|\,\mathbf{y}) = f(\mathbf{x})$. It is easy to verify that all the conditions in the above Assumption 3.2.1 are satisfied with these three choices.

Next, by taking advantage of the proximal operator in the context of centralized optimization, we define the networked proximal operator $\mathbf{nProx}_{\mathcal{X},h_i}(\cdot):\mathbb{R}^{d_i p}\to\mathbb{R}^p$ as,

$$\mathbf{nProx}_{\mathcal{X},h_i}(\mathbf{x}_{j\in\mathcal{N}_i}):=\arg\min_{\mathbf{x}\in\mathcal{X}}\ \Big\{h_i(\mathbf{x})+\frac{1}{2}\cdot\sum_{j\in\mathcal{N}_i}\|\mathbf{x}-\mathbf{x}_j\|^2\Big\}. \tag{3.11}$$

With the help of the defined networked proximal operator, as well as the generic approximation function $g_i(\cdot\,|\,\cdot)$, we now present our NetProx algorithm as below, for $\forall i\in\mathcal{I}$,

$$\begin{aligned}\mathbf{x}_i^{k+1}&=\mathbf{nProx}_{\mathcal{X},\alpha^k g_i}(\mathbf{x}_{j\in\mathcal{N}_i}^k)\\&=\arg\min_{\mathbf{x}\in\mathcal{X}}\ \Big\{g_i(\mathbf{x}\,|\,\mathbf{x}_i^k)+\frac{1}{2\alpha^k}\cdot\sum_{j\in\mathcal{N}_i}\|\mathbf{x}-\mathbf{x}_j^k\|^2\Big\}.\end{aligned} \tag{3.12}$$

The detailed steps of the algorithm are outlined as Algorithm 3.

Algorithm 3: NetProx Algorithm

Data: Given a sequence of the diminishing step-sizes $\{\alpha^k\}_{k\in\mathbb{N}_+}$, each agent i locally initializes its own state $\mathbf{x}_i^0$. Let $k=0$.

while a termination criterion is NOT satisfied **do**

Each agent $i\in\mathcal{I}$ **simultaneously** does

`(S.1)` Receive the information $\mathbf{x}_j^k$ from the neighbors $j\in\mathcal{N}_i$;

`(S.2)` Determine the local approximation function $g_i(\cdot\,|\,\mathbf{x}_i^k)$, perform the networked proximal operator $\mathbf{nProx}_{\mathcal{X},\alpha^k g_i}(\cdot)$, and let

$$\mathbf{x}_i^{k+1}=\mathbf{nProx}_{\mathcal{X},\alpha^k g_i}(\mathbf{x}_{j\in\mathcal{N}_i}^k); \tag{3.13}$$

`(S.3)` Broadcast the updated state $\mathbf{x}_j^{k+1}$ to the neighbors;

`(S.4)` Let $k\leftarrow k+1$ and continue.

end

Before proceeding to the convergence analysis, a few remarks should be made here regarding the proposed NetProx algorithm. First, the algorithm can be implemented in the fully distributed fashion, since each agent only relies on the information received from its neighbors. Second, due to the proximal term and also the fact that the approximation function $g_i(\cdot\,|\,\cdot)$ is assumed to be convex; see Assumption 3.2.1, the minimization problem defined by each $\mathbf{nProx}_{\mathcal{X},\alpha^k g_i}(\cdot)$ is guaranteed to be strongly-convex and thus has the unique solu-

tion. Third, as mentioned before, one can choose among a few approximation functions to implement the NetProx algorithm. However, different choices will result in different computational cost at each iteration, as well as different overall convergence rate of the algorithm; see the detailed convergence analysis in the next section. We shall highlight here that such flexibility offers the great potential to further balance the computation and communication for solving the distributed optimization problem. This is also why we claimed that our NetProx algorithm provides the generalized algorithmic framework.

3.3 Convergence Analysis

In this section, we provide the convergence analysis of the NetProx algorithm. To facilitate the following analysis, let us first introduce some additional notations. Recall that, in Section 3.1, we have defined the new weight matrix $M = D^{-1}A \in \mathbb{R}^{I\times I}$ where D is the diagonal degree matrix and A is the adjacency matrix. It is known that the matrix M is row-stochastic, i.e., $M\mathbf{1}_I = \mathbf{1}_I$. More specifically, let us use $\mathbf{a}_i^\top \in \mathbb{R}^I$ and $\mathbf{m}_i^\top \in \mathbb{R}^I$ to represent the i-th row of the matrices A and M, respectively, and by the definition of the diagonal matrix $D = \text{diag}\{d_1, d_2, \cdots, d_I\}$, it follows that $\mathbf{m}_i = \mathbf{a}_i/d_i$. In addition, we introduce a new vector $\boldsymbol{\epsilon}^k \in \mathbb{R}^{Ip}$, defined as

$$\boldsymbol{\epsilon}^k := \mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^k \tag{3.14}$$

where $\mathbf{x}^k$ corresponds to the agents' aggregated states at the iteration k, and $\mathbf{M} = M \otimes \mathbf{I}_p$ as defined before. Next, we show, by the following lemma, that the norm of this new vector $\boldsymbol{\epsilon}^k$ is bounded by the step-size α^k.

Lemma 3.3.1. *Under Assumptions 1.4.2, 1.4.4 and 3.2.1, let $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$ be the sequence generated by the algorithm (3.12) and $\{\boldsymbol{\epsilon}^k\}_{k\in\mathbb{N}_+}$ be defined as (3.14), then there exists a constant $c_1 > 0$ such that*

$$\|\boldsymbol{\epsilon}^k\| \leq c_1\alpha^k, \quad \forall k \in \mathbb{N}_+. \tag{3.15}$$

Proof. Similar to the state variable $\mathbf{x}^k$, let us denote $\boldsymbol{\epsilon}^k$ as the aggregate of local $\boldsymbol{\epsilon}_i^k$'s, i.e., $\boldsymbol{\epsilon}^k = [(\boldsymbol{\epsilon}_1^k)^\top, (\boldsymbol{\epsilon}_2^k)^\top, \cdots, (\boldsymbol{\epsilon}_I^k)^\top]^\top$. By the definition of $\boldsymbol{\epsilon}^k$, it is straightforward to verify that

$$\boldsymbol{\epsilon}_i^k = \mathbf{x}_i^{k+1} - \mathbf{M}_i^\top \mathbf{x}^k, \tag{3.16}$$

where $\mathbf{M}_i^\top = \mathbf{m}_i^\top \otimes \mathbf{I}_p \in \mathbb{R}^{p \times Ip}$.

According to the iteration (3.12) of the NetProx algorithm, the first order optimality condition guarantees that, for $\forall \mathbf{x} \in \mathcal{X}$,

$$\left(\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k) + \frac{1}{\alpha^k} \cdot \sum_{j \in \mathcal{N}_i} (\mathbf{x}_i^{k+1} - \mathbf{x}_j^k)\right)^\top (\mathbf{x} - \mathbf{x}_i^{k+1}) \geq 0. \tag{3.17}$$

Notice that $\mathbf{M}_i^\top \mathbf{x}^k = \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^k / d_i$, and thus the variable $\mathbf{M}_i^\top \mathbf{x}^k$, as the convex combination of $\mathbf{x}_j^k$'s, must be in the set $\mathcal{X}$. Subsequently, we let $\mathbf{x} = \mathbf{M}_i^\top \mathbf{x}^k$ and rearrange the above inequality (3.17), it yields,

$$\begin{aligned} \|\boldsymbol{\epsilon}_i^k\|^2 &= \|\mathbf{x}_i^{k+1} - \mathbf{M}_i^\top \mathbf{x}^k\|^2 \\ &\leq \frac{\alpha^k}{d_i} \cdot \nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k)^\top (\mathbf{M}_i^\top \mathbf{x}^k - \mathbf{x}_i^{k+1}) \\ &\leq \frac{\alpha^k}{d_i} \cdot \|\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k)\| \cdot \|\mathbf{x}_i^{k+1} - \mathbf{M}_i^\top \mathbf{x}^k\| \\ &= \frac{\alpha^k}{d_i} \cdot \|\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k)\| \cdot \|\boldsymbol{\epsilon}_i^k\|. \end{aligned} \tag{3.18}$$

Next, we note that $\|\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k)\|$ is always bounded, since

$$\begin{aligned} \|\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k)\| &\overset{(3.1.a)}{\leq} \|\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^k)\| + \|\nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^k)\| \\ &\overset{(3.1.b)}{\leq} L_i^g \|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\| + \|\nabla f_i(\mathbf{x}_i^k)\| \\ &\overset{(3.1.c)}{<} \infty, \end{aligned} \tag{3.19}$$

where the inequality (3.1.a) is due to the triangle inequality; (3.1.b) follows from Assumption 3.2.1; and (3.1.c) is due to the compactness of the set $\mathcal{X}$ and the boundedness of the gradient; see Assumption 1.4.2 and inequality (3.9).

As a result of the inequality (3.18) and the boundedness of the term $\|\nabla g_i(\mathbf{x}_i^{k+1} \,|\, \mathbf{x}_i^k)\|$, there must exist a constant $c_1' > 0$ such that

$$\|\boldsymbol{\epsilon}_i^k\| \leq c_1' \alpha^k. \tag{3.20}$$

By the definition of $\boldsymbol{\epsilon}^k$ and letting $c_1 = c_1'\sqrt{I}$, the proof of Lemma 3.3.1 is completed. □

With the help of the above Lemma 3.3.1, we are now ready to bound the consensus error by the following lemma. Note that the consensus error at each iteration k is defined by the quantity $\|\mathbf{x}^k - \boldsymbol{\Pi}\mathbf{x}^k\|$, where $\boldsymbol{\Pi} = (\mathbf{1}_I \boldsymbol{\pi}^\top) \otimes \mathbf{I}_p \in \mathbb{R}^{Ip \times Ip}$ and $\boldsymbol{\pi} \in \mathbb{R}^I$ is the left eigenvector of the weight matrix M corresponding to the eigenvalue 1.

Lemma 3.3.2. *Under Assumptions 1.4.2, 1.4.4, 1.4.8 and 3.2.1, let $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$ be the sequence generated by the algorithm (3.12), then the following inequality holds,*

$$\|\mathbf{x}^k - \boldsymbol{\Pi}\mathbf{x}^k\| \leq c_0(\rho_0)^k \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-1} c_0 c_1 (\rho_0)^{k-1-t} \alpha^t, \tag{3.21}$$

where the constants c_0 and $0 < \rho_0 < 1$ are defined in Lemma 1.4.1.

Proof. It follows by the definition (3.14) of $\boldsymbol{\epsilon}^k$ that

$$\mathbf{x}^{k+1} = \mathbf{M}\mathbf{x}^k + \boldsymbol{\epsilon}^k. \tag{3.22}$$

Let us apply the above equation recursively, it holds that

$$\mathbf{x}^k = (\mathbf{M})^k \mathbf{x}^0 + \sum_{t=0}^{k-1} (\mathbf{M})^{k-1-t} \boldsymbol{\epsilon}^t. \tag{3.23}$$

Now, based on the facts that $\boldsymbol{\pi}$ is the left-eigenvector of M, i.e., $\boldsymbol{\pi}^\top M = \boldsymbol{\pi}^\top$ and the definitions of the matrices $\boldsymbol{\Pi}$ and $\mathbf{M}$, it holds that $\boldsymbol{\Pi}\mathbf{M} = \boldsymbol{\Pi}$ and thus

$$\boldsymbol{\Pi}\mathbf{x}^k = \boldsymbol{\Pi}\mathbf{x}^0 + \sum_{t=0}^{k-1} \boldsymbol{\Pi}\boldsymbol{\epsilon}^t. \tag{3.24}$$

Therefore, it follows that

$$\begin{aligned}\|\mathbf{x}^k - \mathbf{\Pi}\mathbf{x}^k\| &= \left\|\Big((\mathbf{M})^k - \mathbf{\Pi}\Big)\mathbf{x}^0 + \sum_{t=0}^{k-1}\Big((\mathbf{M})^{k-1-t} - \mathbf{\Pi}\Big)\boldsymbol{\epsilon}^t\right\| \\ &\leq \|(\mathbf{M})^k - \mathbf{\Pi}\| \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-1}\|(\mathbf{M})^{k-1-t} - \mathbf{\Pi}\| \cdot \|\boldsymbol{\epsilon}^t\| \\ &\overset{(3.2.a)}{\leq} c_0(\rho_0)^k \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-1} c_0(\rho_0)^{k-1-t} \cdot \|\boldsymbol{\epsilon}^t\| \\ &\overset{(3.2.b)}{\leq} c_0(\rho_0)^k \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-1} c_0 c_1(\rho_0)^{k-1-t}\alpha^t,\end{aligned} \tag{3.25}$$

where the inequality (3.2.a) follows from Lemma 1.4.1 in Chapter 1 and (3.2.b) is due to the above Lemma 3.3.1. □

Subsequently, let us prove the following supporting lemma, which characterizes the key term $\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\|$.

Lemma 3.3.3. *Under Assumptions 1.4.2, 1.4.3, 1.4.4, 1.4.8 and 3.2.1, let $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$ be the sequence generated by the algorithm (3.12), then the following inequality holds,*

$$\begin{aligned}&\sum_{i=1}^{I}\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\| \\ &\leq (L_{\max}^f + L_{\max}^g)\sqrt{I} \cdot \left(2c_0(\rho_0)^{k-1} \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-2} 2c_0 c_1(\rho_0)^{k-2-t}\alpha^t + c_1\alpha^{k-1}\right),\end{aligned} \tag{3.26}$$

where $L_{\max}^f := \max_{1\leq i\leq I} L_i^f$ and $L_{\max}^g := \max_{1\leq i\leq I} L_i^g$.

Proof. Notice that

$$\begin{aligned}&\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\| \\ &\leq \|\nabla f_i(\mathbf{x}_i^k) - \nabla f_i(\mathbf{x}_i^{k-1})\| + \|\nabla f_i(\mathbf{x}_i^{k-1}) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\| \\ &\leq (L_i^f + L_i^g) \cdot \|\mathbf{x}_i^k - \mathbf{x}_i^{k-1}\|,\end{aligned} \tag{3.27}$$

where the last inequality follows from Assumptions 1.4.3 and 3.2.1. As a consequence, it holds that

$$
\begin{aligned}
&\sum_{i=1}^{I} \|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\| \\
&\leq \sum_{i=1}^{I} (L_i^f + L_i^g) \cdot \|\mathbf{x}_i^k - \mathbf{x}_i^{k-1}\| \\
&\overset{(3.3.a)}{\leq} (L_{\max}^f + L_{\max}^g)\sqrt{I} \cdot \|\mathbf{x}^k - \mathbf{x}^{k-1}\| \\
&\overset{(3.3.b)}{=} (L_{\max}^f + L_{\max}^g)\sqrt{I} \cdot \left\| \left((\mathbf{M})^k - (\mathbf{M})^{k-1} \right) \mathbf{x}^0 + \sum_{t=0}^{k-2} \mathbf{M}^{k-2-t}(\mathbf{M} - \mathbf{I}_{Ip})\boldsymbol{\epsilon}^t + \boldsymbol{\epsilon}^{k-1} \right\| \\
&\overset{(3.3.c)}{\leq} (L_{\max}^f + L_{\max}^g)\sqrt{I} \cdot \left(2c_0(\rho_0)^{k-1} \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-2} 2c_0(\rho_0)^{k-2-t} \cdot \|\boldsymbol{\epsilon}^t\| + \|\boldsymbol{\epsilon}^{k-1}\| \right) \\
&\overset{(3.3.d)}{\leq} (L_{\max}^f + L_{\max}^g)\sqrt{I} \cdot \left(2c_0(\rho_0)^{k-1} \cdot \|\mathbf{x}^0\| + \sum_{t=0}^{k-2} 2c_0 c_1 (\rho_0)^{k-2-t} \alpha^t + c_1 \alpha^{k-1} \right),
\end{aligned}
\tag{3.28}
$$

where $(3.3.a)$ is due to the Cauchy-Schwartz inequality; $(3.3.b)$ follows from the dynamics (3.23) of the state $\mathbf{x}^k$; $(3.3.c)$ follows from Lemma 1.4.1 in Chapter 1; and $(3.3.d)$ is based on the diminishing step-sizes and Lemma 3.3.1. □

With the help of the above supporting lemmas, we are now ready to study the convergence of our NetProx algorithm. Before proceeding to the main theorem, let us again introduce some additional notations. We define a real-valued function $\mathbf{f}(\cdot) : \mathbb{R}^{Ip} \to \mathbb{R}$ as,

$$
\mathbf{f}(\mathbf{x}^k) = \sum_{i=1}^{I} f_i(\mathbf{x}_i^k), \tag{3.29}
$$

and thus its gradient $\nabla \mathbf{f}(\mathbf{x}^k) \in \mathbb{R}^{Ip}$ should aggregate all the local gradients as we have defined before. In addition, we use $\bar{\mathbf{x}}^k \in \mathbb{R}^p$ to represent the (weighted) average of the local $\mathbf{x}_i^k$'s, i.e.,

$$
\bar{\mathbf{x}}^k = (\boldsymbol{\pi}^\top \otimes \mathbf{I}_p) \cdot \mathbf{x}^k = \sum_{i=1}^{I} \pi_i \mathbf{x}_i^k. \tag{3.30}
$$

Note that, by the definition of the matrix $\boldsymbol{\Pi}$, it follows that $\boldsymbol{\Pi}\mathbf{x}^k = (\mathbf{1}_I \otimes \mathbf{I}_p) \cdot \bar{\mathbf{x}}^k$.

Now, we are in the position to state the main theorem.

Theorem 3.3.1. *Under Assumptions 1.4.2, 1.4.3, 1.4.4, 1.4.8 and 3.2.1, let* $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$ *be the sequence generated by the algorithm (3.12) and step-size* α^k *be satisfied with i)* $\sum_{k=0}^{\infty}\alpha^k=\infty$*; and ii)* $\sum_{k=0}^{\infty}(\alpha^k)^2<\infty$*, then the following convergence result holds,*

$$\lim_{k\to\infty}\ \|\mathbf{x}_i^k-\mathbf{x}^\star\|=0,\ \forall i\in\mathcal{I}, \tag{3.31}$$

where $\mathbf{x}^\star$ *denotes the optimal solution of Problem (P).*

Proof. By noticing that

$$F(\bar{\mathbf{x}}^k)-F(\mathbf{x}^\star)=\Big(F(\bar{\mathbf{x}}^k)-\mathbf{f}(\mathbf{x}^k)\Big)+\Big(\mathbf{f}(\mathbf{x}^k)-F(\mathbf{x}^\star)\Big), \tag{3.32}$$

let us next the two terms on the right-hand-side of the above (3.32) separately.

First, by the definitions of the vector $\bar{\mathbf{x}}^k$ and the function $\mathbf{g}(\cdot)$, it holds that

$$\begin{aligned}F(\bar{\mathbf{x}}^k)-\mathbf{f}(\mathbf{x}^k)&=\sum_{i=1}^{I}\Big(f_i(\bar{\mathbf{x}}^k)-f_i(\mathbf{x}_i^k)\Big)\\&\overset{(3.4.a)}{\leq}\sum_{i=1}^{I}\nabla f_i(\bar{\mathbf{x}}^k)^\top(\bar{\mathbf{x}}^k-\mathbf{x}_i^k)\\&\overset{(3.4.b)}{\leq}\kappa\cdot\sum_{i=1}^{I}\|\bar{\mathbf{x}}^k-\mathbf{x}_i^k\|\\&\overset{(3.4.c)}{\leq}\kappa\sqrt{I}\cdot\|\mathbf{\Pi}\mathbf{x}^k-\mathbf{x}^k\|,\end{aligned} \tag{3.33}$$

where (3.4.a) is due to the convexity of the functions f_i'; see Assumption 1.4.4, (3.4.b) is due to the boundedness of the gradient; see the inequality (3.9), and (3.4.c) follows from the Cauchy-Schwartz inequality. Further, based on Lemma 3.3.2, it follows that

$$\sum_{t=1}^{k}\alpha^t\|\mathbf{\Pi}\mathbf{x}^t-\mathbf{x}^t\|\leq\Big(\sum_{t=1}^{k}c_0\alpha^t(\rho_0)^t\cdot\|\mathbf{x}^0\|+\sum_{t=1}^{k}\sum_{s=0}^{t-1}c_0c_1(\rho_0)^{t-1-s}(\alpha^s)^2\Big)<\infty. \tag{3.34}$$

Note that the second inequality follows from Lemma 1.4.2 and 1.4.3 in Chapter 1 and the fact that the diminishing step-size α^k satisfies i) $\sum_{k=0}^{\infty}\alpha^k=\infty$; and ii) $\sum_{k=0}^{\infty}(\alpha^k)^2<\infty$.

Next, invoking again the convexity of the objective function $f_i(\cdot)$ yields,

$$
\begin{aligned}
&f_i(\mathbf{x}_i^k) - f_i(\mathbf{x}^\star) \leq \nabla f_i(\mathbf{x}_i^k)^\top (\mathbf{x}_i^k - \mathbf{x}^\star) \\
&= \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})^\top (\mathbf{x}_i^k - \mathbf{x}^\star) + \Big(\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big)^\top (\mathbf{x}_i^k - \mathbf{x}^\star) \\
&\overset{(3.5.a)}{\leq} -\frac{d_i}{\alpha^k}(\mathbf{x}_i^k - \hat{\mathbf{x}}_i^{k-1})^\top (\mathbf{x}_i^k - \mathbf{x}^\star) + \Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\| \cdot \Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\| \\
&= -\frac{d_i}{2\alpha^k}\Big(\|\mathbf{x}_i^k - \mathbf{x}^\star\|^2 + \|\mathbf{x}_i^k - \hat{\mathbf{x}}_i^{k-1}\|^2 - \|\hat{\mathbf{x}}_i^{k-1} - \mathbf{x}^\star\|^2\Big) \\
&\quad + \Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\| \cdot \Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\| \\
&\leq \frac{d_i}{2\alpha^k}\Big(\|\hat{\mathbf{x}}_i^{k-1} - \mathbf{x}^\star\|^2 - \|\mathbf{x}_i^k - \mathbf{x}^\star\|^2\Big) + \Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\| \cdot \Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\| \\
&\overset{(3.5.b)}{\leq} \frac{1}{2\alpha^k}\sum_{j\in\mathcal{N}_i}\Big(\|\mathbf{x}_j^{k-1} - \mathbf{x}^\star\|^2 - \|\mathbf{x}_i^k - \mathbf{x}^\star\|^2\Big) + \Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\| \cdot \Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\|.
\end{aligned}
\tag{3.35}
$$

Note that in $(3.5.a)$, we take advantage of the the inequality (3.17) by letting $\mathbf{x} = \mathbf{x}^\star$ and denote $\hat{\mathbf{x}}_i^{k-1} = \mathbf{M}_i^\top \mathbf{x}^{k-1} = \sum_{j\in\mathcal{N}_i} \mathbf{x}_j^{k-1}/d_i$; in addition, $(3.5.b)$ is due to the facts that $\sum_{j\in\mathcal{N}_i} 1/d_i = 1, \forall i \in \mathcal{I}$ and the convexity of the function of squared norm, i.e.,

$$
\|\hat{\mathbf{x}}_i^{k-1} - \mathbf{x}^\star\|^2 = \Big\|\sum_{j\in\mathcal{N}_i}\frac{1}{d_i}(\mathbf{x}_j^{k-1} - \mathbf{x}^\star)\Big\|^2 \leq \frac{1}{d_i}\cdot\sum_{j\in\mathcal{N}_i}\|\mathbf{x}_j^{k-1} - \mathbf{x}^\star\|^2. \tag{3.36}
$$

Now, let us sum the inequality (3.35) for $\forall i \in \mathcal{I}$, it holds that

$$
\begin{aligned}
\mathbf{f}(\mathbf{x}^k) - F(\mathbf{x}^\star) \leq \frac{1}{2\alpha^k}\Big(&\|\mathbf{x}^{k-1} - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 - \|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2\Big) \\
&+ \sum_{i=1}^I \Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\| \cdot \Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\|,
\end{aligned}
\tag{3.37}
$$

where the norm $\|\cdot\|_{\mathbf{D}}^2$ is defined as $\|\mathbf{x}\|_{\mathbf{D}}^2 = \mathbf{x}^\top\mathbf{D}\mathbf{x}$. Further, by Lemma 3.3.3, one can have

$$
\begin{aligned}
&\sum_{t=1}^k \alpha^t \sum_{i=1}^I \Big\|\nabla f_i(\mathbf{x}_i^t) - \nabla g_i(\mathbf{x}_i^t \,|\, \mathbf{x}_i^{t-1})\Big\| \cdot \Big\|\mathbf{x}_i^t - \mathbf{x}^\star\Big\| \\
&\leq (L_{\max}^f + L_{\max}^g)\sqrt{I}\cdot\Big\|\mathbf{x}_i^t - \mathbf{x}^\star\Big\| \\
&\quad\cdot\Big(\sum_{t=1}^k 2c_1(\rho_0)^{t-1}\alpha^t\cdot\|\tilde{\mathbf{x}}^0\| + \sum_{t=1}^k\sum_{s=0}^{t-2} 2c_1c_1(\rho_0)^{t-2-s}(\alpha^s)^2 + \sum_{t=1}^k c_1(\alpha^{t-1})^2\Big) \\
&< \infty.
\end{aligned}
\tag{3.38}
$$

Note that the last inequality is due to the fact $\|\mathbf{x}_i^t - \mathbf{x}^\star\| < \infty$ owing to the compactness of the feasible set $\mathcal{X}$, as well as Lemmas 1.4.2 and 1.4.3 in Chapter 1.

Consequently, summing up the above inequalities (3.33) and (3.37) and rearranging it gives,

$$\begin{aligned}
&\|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot \mathbf{x}^\star\|_{\mathbf{D}}^2 \\
&\leq \|\mathbf{x}^{k-1} - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot \mathbf{x}^\star\|_{\mathbf{D}}^2 - 2\alpha^k \cdot \Big(F(\bar{\mathbf{x}}^k) - F(\mathbf{x}^\star)\Big) \\
&\quad \underbrace{+ 2\alpha^k \cdot \Big(\sum_{i=1}^{I} \Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\| \cdot \Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\| + \kappa\sqrt{I}\cdot \|\mathbf{\Pi}\mathbf{x}^k - \mathbf{x}^k\|\Big)}_{\Xi^k}.
\end{aligned} \tag{3.39}$$

Let us denote the last term on the right side as Ξ^k, according to the inequalities (3.34) and (3.38), we can have that the sequence $\{\Xi^k\}_{k\in\mathbb{N}_+}$ must be summable, i.e., $\sum_{k=1}^{\infty}\Xi^t < \infty$, when the step-size α^k satisfies the two conditions. Therefore, based on Lemma 1.4.4 in Chapter 1 and the fact that $F(\bar{\mathbf{x}}^k) - F(\mathbf{x}^\star) \geq 0$, it holds that,

$$\sum_{t=1}^{k} \alpha^t \Big(F(\bar{\mathbf{x}}^t) - F(\mathbf{x}^\star)\Big) < \infty, \tag{3.40}$$

and there must exist a constant $\delta \geq 0$ such that,

$$\lim_{k\to\infty} \|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot \mathbf{x}^\star\|_{\mathbf{D}}^2 = \delta. \tag{3.41}$$

Now, recall again that the step size has $\sum_{t=1}^{\infty}\alpha^t = \infty$, thus it follows from (3.40) that $\liminf_{k\to\infty} F(\bar{\mathbf{x}}^k) = F(\mathbf{x}^\star)$ and further implies that there exists a subsequence $\{\bar{\mathbf{x}}^{k_n}\}_{n\in\mathbb{N}_+}$ such that

$$\lim_{n\to\infty} F(\bar{\mathbf{x}}^{k_n}) = F(\mathbf{x}^\star). \tag{3.42}$$

As a consequence of the compactness of the feasible set $\mathcal{X}$, there must exist a subsequence $\{\bar{\mathbf{x}}^{k_l}\}_{l\in\mathbb{N}_+} \subseteq \{\bar{\mathbf{x}}^{k_n}\}_{n\in\mathbb{N}_+}$ such that $\lim_{l\to\infty}\bar{\mathbf{x}}^{k_l} = \mathbf{x}^\star$.

Then, in order to prove the statement in the theorem, it only remains to show that $\delta = 0$ in (3.41). Notice that

$$\begin{aligned}
\delta &= \liminf_{k\to\infty} \ \|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot \mathbf{x}^\star\|_{\mathbf{D}}^2 \\
&= \liminf_{k\to\infty} \ \sum_{i=1}^{I} d_i \cdot \|\mathbf{x}_i^k - \mathbf{x}^\star\|^2 \\
&\leq \liminf_{k\to\infty} \ \sum_{i=1}^{I} 2d_i \cdot \Big(\|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\|^2 + \|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2\Big) \\
&= \liminf_{k\to\infty} \ \sum_{i=1}^{I} 2d_i \cdot \|\bar{\mathbf{x}}^k - \mathbf{x}^\star\|^2 = 0.
\end{aligned} \tag{3.43}$$

Note that the second last equality is due to the the super-additivity property of the limit inferior and the fact that $\lim_{k\to\infty} \|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\| = 0$ based on Lemma 3.3.2 and Lemma 1.4.1 in Chapter 1. Therefore, the positive constant δ must be zero and the proof is complete. □

We shall remark that the above Theorem 3.3.1 states the convergence of the generalized NetProx framework, i.e., regardless of the choice of the approximation function $g_i(\cdot\,|\,\cdot)$ as long as it satisfies Assumption 3.2.1. In addition, under the condition of the step-sizes, i.e., $\sum_{k=0}^{\infty}\alpha^k = \infty$ and $\sum_{k=0}^{\infty}(\alpha^k)^2 < \infty$, the convergence of algorithm is characterized by $\mathbf{x}_i^k \to \mathbf{x}^\star$, which also implies the consensus of the solutions. Next, in order to provide the convergence rate of the NetProx algorithm, we study three different choices of the approximation functions separately: i) the original function; ii) the second order approximation; and iii) the linear approximation. Note that here the convergence of algorithm is described in the ergodic sense with respect to the weighted averaged state of all agents; see details in Theorem 3.3.2, and the step-size is specified as $\alpha^k = 1/\sqrt{k}$ in the following analysis.

Theorem 3.3.2. *Under Assumptions 1.4.2, 1.4.3, 1.4.4, 1.4.8 and 3.2.1, let $\{\mathbf{x}^k\}_{k\in\mathbb{N}_+}$ be the sequence generated by the algorithm (3.12) with step-size specified as $\alpha^k = 1/\sqrt{k}$, then convergence rate can be given as $F(\tilde{\mathbf{x}}^k) - F(\mathbf{x}^\star) = \mathcal{O}\Big(1/\sqrt{k}\Big)$, where $\tilde{\mathbf{x}}^k \in \mathbb{R}^p$ is defined as $\tilde{\mathbf{x}}^k = (1/k)\cdot\sum_{t=1}^{k}\bar{\mathbf{x}}^t$ and $\bar{\mathbf{x}}^t$ is given by (3.30). In particular, considering the three specific choices of the approximation function $g_i(\cdot\,|\,\cdot)$, one can have that i) $F(\tilde{\mathbf{x}}^k) - F(\mathbf{x}^\star) \sim \mathcal{O}\Big(\gamma_0/\sqrt{k}\Big)$ for the original function, i.e., $g_i(\mathbf{x}\,|\,\mathbf{x}_i^k) = f_i(\mathbf{x})$; ii) $F(\tilde{\mathbf{x}}^k) - F(\mathbf{x}^\star) \sim \mathcal{O}\Big(\gamma_0/\sqrt{k} +$*

$\gamma_1/\sqrt{k}\big)$ *for the linear approximation; and iii)* $F(\tilde{\mathbf{x}}^k) - F(\mathbf{x}^\star) \sim \mathcal{O}\big(\gamma_0/\sqrt{k} + \ln(k)/k\big)$ *for the second order approximation.*

Proof. Let us recall the two inequalities (3.33) and (3.37), summing them together yields,

$$\begin{aligned}\Big(F(\bar{\mathbf{x}}^k) - F(\mathbf{x}^\star)\Big) \leq\ & \underbrace{\frac{1}{2\alpha^k}\Big(\|\mathbf{x}^{k-1} - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 - \|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2\Big)}_{\mathcal{T}_1^k} \\ & + \underbrace{\kappa\sqrt{I}\cdot\|\mathbf{\Pi}\mathbf{x}^k - \mathbf{x}^k\|}_{\mathcal{T}_2^k} + \underbrace{\sum_{i=1}^{I}\Big\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\Big\|\cdot\Big\|\mathbf{x}_i^k - \mathbf{x}^\star\Big\|}_{\mathcal{T}_3^k}.\end{aligned} \tag{3.44}$$

Now, let us investigate the three terms on the right hand side of (3.44) separately. First of all, the summation of $\mathcal{T}_1^k$'s is given by

$$\begin{aligned}\sum_{t=1}^{k}\mathcal{T}_1^t &= \sum_{t=1}^{k}\frac{1}{2\alpha^t}\Big(\|\mathbf{x}^{t-1} - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 - \|\mathbf{x}^t - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2\Big) \\ &= \frac{1}{2}\cdot\sum_{t=1}^{k-1}\Big(\frac{1}{\alpha^{t+1}} - \frac{1}{\alpha^t}\Big)\cdot\|\mathbf{x}^t - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 \\ &\quad + \frac{1}{2\alpha^1}\cdot\|\mathbf{x}^0 - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 - \frac{1}{2\alpha^k}\cdot\|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 \\ &\leq \frac{\beta}{2}\cdot\Big(\frac{1}{\alpha^1} + \sum_{t=1}^{k-1}\Big(\frac{1}{\alpha^{t+1}} - \frac{1}{\alpha^t}\Big)\Big) = \frac{\beta}{2\alpha^k} = c_3\sqrt{k}.\end{aligned} \tag{3.45}$$

Note that, in the first inequality of (3.45), we apply the fact that there must exist some constant $\beta > 0$ such that $\|\mathbf{x}^k - (\mathbf{1}_I \otimes \mathbf{I}_p)\cdot\mathbf{x}^\star\|_{\mathbf{D}}^2 \leq \beta$, $\forall k \in \mathbb{N}_+$ due to the compactness of the feasible set $\mathcal{X}$; and in the last equality, we let $c_3 = \beta/2$ and use the fact $\alpha^k = 1/\sqrt{k}$. Second, according to Lemma 3.3.2, it holds that

$$\begin{aligned}\sum_{t=1}^{k}\mathcal{T}_2^t &\leq \kappa\sqrt{I}\cdot\sum_{t=1}^{k}\Big(c_0(\rho_0)^t\cdot\|\mathbf{x}^0\| + c_0c_1\cdot\sum_{s=0}^{t-1}(\rho_0)^{t-1-s}\alpha^s\Big) \\ &\leq \kappa c_0\sqrt{I}\cdot\frac{\rho_0}{1-\rho_0}\cdot\|\mathbf{x}^0\| + c_0c_1\kappa\sqrt{I}\cdot\sum_{t=1}^{k}c_4'\alpha^{\lfloor t/2\rfloor} \\ &\leq c_4\cdot\sum_{t=1}^{k}\frac{1}{\sqrt{t}}.\end{aligned} \tag{3.46}$$

Note that the second inequality is due to the following fact; see the same argument in the proof of Lemma 1 in [97],

$$\sum_{s=0}^{t-1}(\rho_0)^{t-1-s}\alpha^s = \mathcal{O}(\alpha^{\lfloor t/2\rfloor}), \tag{3.47}$$

and thus there must have some constant $c_4' > 0$ such that $\sum_{s=0}^{t-1}(\rho_0)^{t-1-s}\alpha^s \leq c_4'\alpha^{\lfloor t/2\rfloor}$; and in the last inequality of (3.46), the existence of the constant $c_4 > 0$ is guaranteed by $\alpha^k = 1/\sqrt{k}$.

Lastly, for the term $\mathcal{T}_3^k$, we need to take into account the three different types of the approximation functions. First, when the original $f_i(\cdot)$ is adopted as the approximation function, it is easy to verify that $\mathcal{T}_3^k = 0,\ \forall k \in \mathbb{N}_+$ since $\nabla f_i(\mathbf{x}_i^k) = \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})$. Second, when the linear approximation function is applied, i.e., $\nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1}) = \nabla f_i(\mathbf{x}_i^{k-1})$, then it holds that

$$\begin{aligned}
\sum_{t=1}^{k}\mathcal{T}_{3,L}^t &= \sum_{t=1}^{k}\sum_{i=1}^{I}\left\|\nabla f_i(\mathbf{x}_i^t) - \nabla f_i(\mathbf{x}_i^{t-1})\right\| \cdot \left\|\mathbf{x}_i^t - \mathbf{x}^\star\right\| \\
&\leq \beta \cdot \sum_{t=1}^{k}\sum_{i=1}^{I}\left\|\nabla f_i(\mathbf{x}_i^t) - \nabla f_i(\mathbf{x}_i^{t-1})\right\| \\
&\leq \beta L_{\max}^f \sqrt{I} \cdot \sum_{t=1}^{k}\|\mathbf{x}^t - \mathbf{x}^{t-1}\| \\
&\leq c_5 \sum_{t=1}^{k}\frac{1}{\sqrt{t}}.
\end{aligned} \tag{3.48}$$

Note that here we add the subscript L to denote the term $\mathcal{T}_{3,L}^k$ which corresponds to the linear approximation function, and the last inequality can be obtained by following the proofs of Lemma 3.3.3; see inequality (3.28), as well as the inequality (3.46). Likewise, when the second-order approximation function is applied, we use the subscript Q to denote the term $\mathcal{T}_{3,Q}^k$. Notice that, in this case, the gradient of the approximation function is given by

$$\nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1}) = \nabla f_i(\mathbf{x}_i^{k-1}) + \nabla^2 f_i(\mathbf{x}_i^{k-1})(\mathbf{x}_i^k - \mathbf{x}_i^{k-1}), \tag{3.49}$$

and according to the Taylor's Theorem, there must exist a constant $c_6' > 0$ such that

$$\left\|\nabla f_i(\mathbf{x}_i^k) - \nabla g_i(\mathbf{x}_i^k \,|\, \mathbf{x}_i^{k-1})\right\| \leq c_6' \cdot \|\mathbf{x}_i^k - \mathbf{x}_i^{k-1}\|^2, \quad \forall i \in \mathcal{I}. \tag{3.50}$$

Therefore, following the same path as the inequality (3.48), it holds that there exists a constant $c_6 > 0$ such that

$$\sum_{t=1}^{k} \mathcal{T}_{3,Q}^{t} \leq c_6' \beta \cdot \sum_{t=1}^{k} \|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 \leq c_6 \sum_{t=1}^{k} \frac{1}{t}. \tag{3.51}$$

Now, we are ready to prove the statement in the theorem. Due to the definition of the state $\tilde{\mathbf{x}}^k$ and also the convexity of the function $F(\cdot)$, it follows that

$$\begin{aligned} F(\tilde{\mathbf{x}}^k) - F(\mathbf{x}^\star) &\leq \frac{1}{k} \cdot \sum_{t=1}^{k} \Big(F(\bar{\mathbf{x}}^t) - F(\mathbf{x}^\star) \Big) \\ &\leq \frac{1}{k} \cdot \sum_{t=1}^{k} \Big(\mathcal{T}_1^{\,t} + \mathcal{T}_2^{\,t} + \mathcal{T}_3^{\,t} \Big) \\ &\leq \frac{c_3}{\sqrt{k}} + \frac{c_4 \cdot \sum_{t=1}^{k} 1/\sqrt{t}}{k} + \frac{\sum_{t=1}^{k} \mathcal{T}_3^{\,t}}{k} \\ &= \mathcal{O}\Big(\frac{\gamma_0}{\sqrt{k}} \Big) + \frac{\sum_{t=1}^{k} \mathcal{T}_3^{\,t}}{k}, \end{aligned} \tag{3.52}$$

where the last inequality is due to the fact that $\sum_{t=1}^{k} 1/\sqrt{t} = \mathcal{O}(\sqrt{k})$. Furthermore, when the original function is considered, one can have that $(1/k) \cdot \sum_{t=1}^{k} \mathcal{T}_3^{\,t} = 0$; when the linear approximation function is considered, $\mathcal{T}_3^{\,t}$ will be replaced by $\mathcal{T}_{3,L}^{t}$, and one can have that $(1/k) \cdot \sum_{t=1}^{k} \mathcal{T}_{3,L}^{t} = \mathcal{O}\Big(\gamma_1/\sqrt{k}\Big)$; at last, when the second order approximation function is considered, $\mathcal{T}_3^{\,t}$ is replaced by $\mathcal{T}_{3,Q}^{t}$, and one can have that $(1/k) \cdot \sum_{t=1}^{k} \mathcal{T}_{3,Q}^{t} = \mathcal{O}\Big(\ln(k)/k\Big)$. Therefore, the proof is complete. □

Based on the above Theorem 3.3.2, it can be concluded that, although the overall convergence of the NetProx algorithm is at the rate of $\mathcal{O}(1/\sqrt{k})$ when the step-size is specified as $\alpha^k = 1/\sqrt{k}$. yet different choices of the approximation functions will also result in slightly difference with respect to the performance of the algorithm. In general, high order approximation functions can help to accelerate the convergence. but at the price of more computational cost at each iteration. This is also why we claimed in Section 3.2 that our NetProx algorithm offers flexibility to balance the computation and communication for solving the distributed optimization problem.

4. RESILIENT DISTRIBUTED MIN-MAX OPTIMIZATION UNDER NETWORK COMMUNICATION ATTACKS

This chapter studies a special instance of the distributed optimization problem (P) – the distributed min-max optimization; in particular, we focus on the resilience of algorithm against network communication attacks. More precisely, our algorithm builds on two crucial components: i) a resilient convex combination scheme which helps eliminate the malicious information injected by the unidentifiable communication attacks; and ii) a consensus-based distributed algorithm which solves the min-max optimization over time-varying unbalanced directed graphs. We show that, under reasonable assumptions, e.g., the attacked communication channels can be recovered within a certain time-window, the proposed algorithm converges to the exact global optimal solution which involves every attacked/non-attacked agent within the network. This result is primarily different from the existing relevant works whose the objective only includes the local cost functions at the non-attacked agents.

4.1 Problem Statement

As distinct from the general distributed optimization Problem (P) which has been considered in the previous chapters, in this chapter we focus on a special instance, i.e., the distributed min-max optimization formulated as follows,

$$\min_{\mathbf{x}\in\mathcal{X}} \quad G(\mathbf{x}) := \max_{1\leq i\leq I} \quad f_i(\mathbf{x}). \tag{4.1}$$

Here, each $f_i(\cdot)$ is also the local objective function which is assumed to satisfy the general assumptions in Chapter 1 and supposed to be privately maintained by the corresponding agent $i \in \mathcal{I}$. However, the overall objective now is not to solve for the minimizer of the summation of the local $f_i(\cdot)$'s, but to consider the maximum one among all agents. In fact, such a min-max problem is closely related to the robust optimization [98]–[100], in which the potential uncertainties are sampled via a set of scenarios and the worst case among them is optimized to guaranteed a robust solution. More specifically, in order facilitate the computational tasks, the distributed min-max optimization spreads all the sampled scenarios

into a network of computing facilities, so that each node only needs to deal with a few scenarios and thus the computational cost is significantly reduced compared to the centralized frameworks. It is noteworthy that such distributed min-max problem has also attracted a great amount of attention among researchers; see e.g., [101]–[104] and the references herein.

Despite of the great success of the distributed algorithms reviewed in Chapter 1, we shall emphasize that almost all of the existing methods require a reliable underlying network which always maintains the perfect communication among different agents. Motivated by the emerging studies of cyber-security issues in many real-world applications, see e.g., [105]–[107], it would be a natural question to ask if the existing algorithms could be resilient against the potential network attacks. Unfortunately, two recent papers [20], [21] essentially provided the negative answers to this open question. While it is theoretically proved in [21] that the distributed optimization in general is not solvable (see Theorem 1) if the network attack is present, the authors in [20] show, by a simple counter-example, that any existing distributed algorithm can be easily disrupted by a single attack. In the following, we adapt the simple example in [20] under our distributed min-max problem setting.

Example 1: Without loss of any generality, let us suppose that the first agent ($i = 1$) is under attack. Consider that the attacked agent is allowed to behave however it wishes, and thus is capable of arbitrarily choosing a fake local function $\bar{f}_1(\cdot) : \mathbb{R}^p \to \mathbb{R}$ such that, at the specific point $\bar{\mathbf{x}} \in \mathcal{X}$, it has $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathcal{X}} \bar{f}_1(\mathbf{x})$ and $\bar{f}_1(\bar{\mathbf{x}}) \geq \max_{2\leq i\leq I} \min_{\mathbf{x}\in\mathcal{X}} f_i(\mathbf{x})$. Now, one can easily verify that the specific point $\bar{\mathbf{x}}$ would be the global optimal solution when the network is attacked. Furthermore, due to the arbitrariness of the fake function $\bar{f}_1(\cdot)$, it can be concluded that the attack of this agent can manipulate the global minimizer to any point $\bar{\mathbf{x}}$ in the set $\mathcal{X}$ and thus fake any existing distributed algorithm.

One should notice that an assumption was implicitly made to valid the example; that is, once some agent i is attacked, it will never be recovered during the whole optimization process. As a consequence of this assumption, it is clear that the global optimality has no way to be obtained due to the disappearance of the indispensable local information $f_i(\cdot)$. Nevertheless, starting from a practical perspective, it is reasonable to believe that the real information can be sent out during some time-slot even if the attacks are present within the network. Motivated by this, in order to cope with the vulnerability resulted from the

loss of local cost functions, we consider the following network attack scenario: the targets of attack are changing with time, but every attacked agent can be recovered at least once during some certain time-window. More precisely, let us denote the set of attacked agents as $\mathcal{A}^k \subset \mathcal{I}$ at each discrete time-step $k \in \mathbb{N}_+$, and thus the set $\bar{\mathcal{A}}^k := \mathcal{I} \setminus \mathcal{A}^k$ contains all non-attacked agents. It is assumed that each agent i has no knowledge about which neighbors are under attack, but can access an integer $a_i^k \in \mathbb{N}_+$ at each time-step k which upper bounds the number of the attacked neighbors, i.e., $|\mathcal{A}_i^k| \leq a_i^k$ where $\mathcal{A}_i^k := \mathcal{N}_i \cap \mathcal{A}^k$. Note that, in this chapter, we exclude the self-loop in the set of neighborhood $\mathcal{N}_i$. To ensure the recovery of each attacked agent within some time-window, we also need the following condition regarding the set $\bar{\mathcal{A}}^k$.

Assumption 4.1.1. *There must exist a positive integer $T \in \mathbb{N}_+$ such that*

$$\bigcup_{t=k}^{T+k-1} \bar{\mathcal{A}}^t = \mathcal{I}, \quad \forall k \in \mathbb{N}_+. \tag{4.2}$$

Furthermore, we shall remark that the convergence of any distributed algorithm can be easily destroyed by a direct manipulation of the agents' local states. For instance, even if the algorithm has already driven all agents to the desired optimal solution at some certain time-step k, i.e., $\mathbf{x}_i^k = \mathbf{x}^\star$, $\forall i \in \mathcal{I}$, then a simple change of any single agent's local state $\mathbf{x}_i^k$ would immediately disrupt the convergence. On this basis, to prevent the direct manipulation of agents' local states, we will need the following additional assumption.

Assumption 4.1.2. *It is assumed that the network attacks occur only in the communication channels. That is, if the agent is attacked, then only the send-out information will be manipulated, but the local information for itself is still reliable.*

Note that the above Assumption 4.1.2 is not against the notion of byzantine attack; it is only used to ensure that each agent can always trust the information maintained by itself. In the following, we show, by another example, that the problem (P) does not become immediately trivial, even with both Assumptions 4.1.1 and 4.1.2 satisfied.

Example 2: Without loss of any generality, let us suppose that the first two agents are under attack periodically, i.e., the agent $i = 1$ is attacked at every odd time-step and the

agent $i = 2$ is attacked at every even time-step. It is easy to verify that Assumption 4.1.1 is satisfied (with $T = 2$) under such an attack scenario. Consider that these two agents, when attacked, choose their fake local functions $\bar{f}_1(\cdot)$ and $\bar{f}_2(\cdot)$ such that, at the specific point $\bar{\mathbf{x}} \in \mathcal{X}$, it has i) $\bar{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathcal{X}} \bar{f}_i(\mathbf{x})$ for $i = 1, 2$; ii) $\bar{f}_1(\bar{\mathbf{x}}) \geq \max_{i\neq 1, i\in\mathcal{I}} \min_{\mathbf{x}\in\mathcal{X}} f_i(\mathbf{x})$; and iii) $\bar{f}_2(\bar{\mathbf{x}}) \geq \max_{i\neq 2, i\in\mathcal{I}} \min_{\mathbf{x}\in\mathcal{X}} f_i(\mathbf{x})$. Under these three conditions, it can be concluded that any existing distributed algorithm will drive all the non-attacked agents to converge to the fake optimal solution $\bar{\mathbf{x}}$ and thus is disabled by the periodical attacks.

Provided the vulnerability of the existing distributed algorithms as shown in the preceding Example 1 and 2, both [21] and [20] compromise the global objective to some extent while designing their solution methods. In particular, they completely give up the agents once attacked and only take into account the cost functions of non-attacked agents. Moreover, the optimization problems considered in both [21] and [20] merely focus on the scalar objective functions. It is still unknown how to extend their results into the more general local cost functions. On the contrary, the resilient algorithm presented in this chapter considers the generic objective function as shown in the problem (4.1), and importantly, we care about every single agent within the network no matter attacked or non-attacked. We show that, under some reasonable (arguably necessary) assumptions; see details in the following sections, our algorithm can converge to the exact global optimal solution $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} G(\mathbf{x})$ which counts all local cost functions. This makes our work significantly different from the existing ones.

4.2 Resilient Convex Combination

Let us denote each agent's local state as $\mathbf{x}_i^k \in \mathbb{R}^p$, and assume that all the local states $\mathbf{x}_i^k$'s are in general positions[1], at each time-step k. The problem of interest in this section is to design an appropriate mechanism which helps each agent i to achieve a resilient convex combination of the information received from only the non-attacked neighbors. Note that here the main challenge comes from the fact that each agent communicates with all the

[1]↑A set of points in the space $\mathbb{R}^p$ is said to be in the general position if no hyperplane of dimension $p-1$ or less contains more than p points. Note that, as stated in [108], any randomly drawn set of p points will be in general position with probability one.

attacked and non-attacked neighbors and is usually unable to distinguish the malicious ones from all the received information. Next, we present the scheme proposed in [109], which is able to achieve the resilient convex combination without the need to identify the attacked neighbors.

Suppose that each agent i knows the number of its neighbors $d_i = |\mathcal{N}_i|$, as well as the upper bound a_i^k for the number of the attacked ones. It is straightforward to see that $a_i^k \leq d_i$, for $\forall i \in \mathcal{I}$ and $k \in \mathbb{N}_+$. Then, the resilient convex combination scheme performs the following three steps at each time k:

Step 1) Construct the following set $\mathcal{S}_i^k$ with each element being a subset of the information received from the neighbors,

$$\mathcal{S}_i^k := \Big\{ \mathcal{X} \mid \mathcal{X} \subseteq \{\mathbf{x}_j^k,\ j \in \mathcal{N}_i\},\ |\mathcal{X}| = d_i - a_i^k \Big\}. \tag{4.3}$$

Note that there are totally $\binom{d_i}{d_i - a_i^k}$ possibilities of the subset $\mathcal{X}$, i.e., $|\mathcal{S}_i^k| = \binom{d_i}{d_i - a_i^k}$, as a consequence, we also denote the set as $\mathcal{S}_i^k = \Big\{ \mathcal{X}_i^k(s), s = 1, 2, \cdots, S_i^k \Big\}$ where $S_i^k = \binom{d_i}{d_i - a_i^k}$.

Step 2) Generate the convex hull of the set $\mathcal{X}_i^k(s) \cup \mathbf{x}_i^k$, denoted as $\mathcal{H}\Big(\mathcal{X}_i^k(s) \cup \mathbf{x}_i^k\Big)$, for each $s = 1, 2, \cdots, S_i^k$, and obtain their intersection $\mathcal{C}_i^k \subset \mathbb{R}^p$, i.e.,

$$\mathcal{C}_i^k := \bigcap_{s=1}^{S_i^k} \mathcal{H}\Big(\mathcal{X}_i^k(s) \cup \mathbf{x}_i^k\Big). \tag{4.4}$$

Step 3) Choose an arbitrary point $\mathbf{z}_i^k \in \mathcal{C}_i^k$ as the resilient convex combination of the information received from only the non-attacked neighbors.

It should be highlighted that the intersection set $\mathcal{C}_i^k$ must be non-empty, since it is always true by definition that $\mathbf{x}_i^k \in \mathcal{C}_i^k$; in addition, the point $\mathbf{x}_i^k$ also constitutes a trivial resilient convex combination, since each agent i can always trust its local state (see Assumption 4.1.2). Nevertheless, in the more general case, besides the trivial point $\mathbf{x}_i^k$, the set $\mathcal{C}_i^k$ should also contain other points, as shown in Fig. 4.1. In this example, the first agent ($i = 1$) receives information from all its neighbors, and knows that there is at most one of them being under attack. The intersection $\mathcal{C}_1$ of all generated convex hulls is shown as the red line. An arbitrary point in $\mathcal{C}_1$ can be always viewed as a convex combination of the information received from

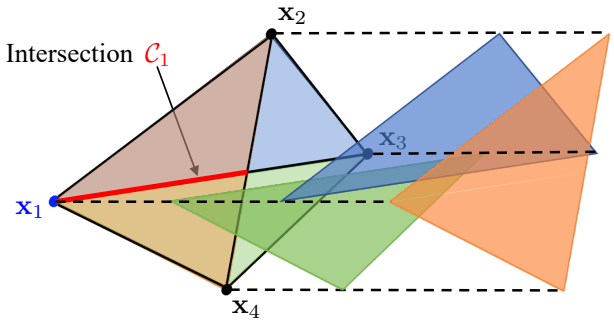


Figure 4.1. Illustration of the resilient convex combination

only the non-attacked agents, no matter which one is under attack. More details will be discussed in Remark 4.2.2 and also in the next section.

Now, based on the above Steps 1) – 3) as well as the results in [109] (see Lemmas 1 – 3), it is known that an arbitrary point $\mathbf{z}_i^k$ from the set $\mathcal{C}_i^k$ is a valid resilient convex combination, i.e., combining the information received from only the non-attacked neighbors. Therefore, let us express the point $\mathbf{z}_i^k$ in the following convex combination form,

$$\mathbf{z}_i^k = \sum_{j \in \mathcal{N}_i^+ \backslash \mathcal{A}_i^k} \beta_{ij}^k \mathbf{x}_j^k \tag{4.5}$$

where $\mathcal{N}_i^+ := \mathcal{N}_i \cup \{i\}$ and the coefficients β_{ij}^k's have $\beta_{ij}^k \geq 0$ and $\sum_{j \in \mathcal{N}_i^+ \backslash \mathcal{A}_i^k} \beta_{ij}^k = 1$. In this sense, the selection of the specific point $\mathbf{z}_i^k$ from the set $\mathcal{C}_i^k$ is equivalent to deciding the convex combination coefficients β_{ij}^k's. In fact, such a group of coefficients β_{ij}^k's can be implicitly determined by solving the following linear programming problem,

$$\max_{\mathbf{z},\, \boldsymbol{\beta}(s),\, \kappa} \quad \kappa \tag{4.6a}$$

$$\text{s. t.} \quad \boldsymbol{\beta}(s)^\top \hat{\mathbf{x}}_i^k(s) = \mathbf{z}, \tag{4.6b}$$

$$\mathbf{1}^\top \boldsymbol{\beta}(s) = 1, \tag{4.6c}$$

$$\boldsymbol{\beta}(s) - \kappa \cdot \mathbf{1} \geq 0, \ \forall s = 1, 2, \cdots, S_i^k. \tag{4.6d}$$

Note that here $\boldsymbol{\beta}(s) \in \mathbb{R}^{d_i - a_i^k + 1}$ is a column vector and the corresponding $\hat{\mathbf{x}}_i^k(s) \in \mathbb{R}^{p(d_i - a_i^k + 1)}$ concatenates all states in the set $\mathcal{X}_i^k(s) \cup \mathbf{x}_i^k$. By convention, we let $\mathbf{x}_i^k$ always be the first

component in each $\hat{\mathbf{x}}_i^k(s)$. A few remarks should be added regarding the linear programming problem (4.6).

Remark 4.2.1. *Due to the fact that $\mathbf{x}_i^k$ is always in the set $\mathcal{C}_i^k$, it is straightforward to see that the optimization problem (4.6) must be feasible, as a trivial feasible solution is $\mathbf{z} = \mathbf{x}_i^k$, $\kappa = 0$ and $\boldsymbol{\beta}(s) = \mathbf{e}_1$, $\forall s = 1, 2, \cdots, S_i^k$ where $\mathbf{e}_1 \in \mathbb{R}^{d_i - a_i^k + 1}$ is the first column of the identity matrix. In addition, consider that the number of attacked neighbors of the i-th agent is upper bounded by the integer a_i^k, thus there exists at least one subset $\mathcal{X}_i^k(s^\star) \in \mathcal{S}_i^k$ or equivalently $\hat{\mathbf{x}}_i^k(s^\star)$, which contains only the states from the non-attacked neighbors. As a consequence, the corresponding group of coefficients $\boldsymbol{\beta}(s^\star)$ can be used to express the resilient convex combination in the form of (4.5).*

Remark 4.2.2. *In the more desired cases, it is expected that the resilient convex combination coefficients $\boldsymbol{\beta}(s^\star)$ should contain as many non-zero components as possible, so that the agent can receive more information from the non-attacked neighbors. This is also the reason why we would like to maximize the minimum component of $\boldsymbol{\beta}(s)$ for all $s = 1, 2, \cdots, S_i^k$ in the optimization problem (4.6). Most ideally, the obtained optimal solution $\kappa^\star$ should be strictly greater than zero. As such, it is guaranteed that $\boldsymbol{\beta}(s^\star) > 0$ and thus the i-th agent truly integrates the information from $d_i - a_i^k$ non-attacked neighbors. We postpone the discussion of the conditions which ensures $\kappa^\star > 0$ to the next section; see Proposition 4.3.1 in Section 4.3.2.*

To sum up, let us represent the resilient convex combination scheme for each individual agent i as an abstract operator $\mathcal{R}_i(\cdot) : \prod_{j \in \mathcal{N}_i} \mathbb{R}^p \to \mathbb{R}^p$, and outline the operations as the following Algorithm 4.

Algorithm 4: Resilient Convex Combination $\mathcal{R}_i(\cdot)$

Data: Require the number of the neighbors d_i and upper bound of the number of attacked neighbors a_i^k.

`(S.1)` Receive the states $\mathbf{x}_j^k$ from all attacked and non-attacked neighbors $j \in \mathcal{N}_i$;

`(S.2)` Generate the subset $\mathcal{X}_i^k(s) \subseteq \{\mathbf{x}_j^k,\ j \in \mathcal{N}_i\}$ such that $|\mathcal{X}_i^k(s)| = d_i - a_i^k$;

`(S.3)` Solve the linear program (4.6) and output the obtained solution $\mathbf{z}_i^k$.

4.3 Resilient Distributed Optimization

Equipped with the resilient convex combination operator $\mathcal{R}_i(\cdot)$, in this section, we develop our distributed solution method to solve the original problem (4.1). Before that, let us first introduce our second building block which is originally proposed in [110] – consensus-based distributed optimization algorithm.

4.3.1 Distributed Optimization over Time-Varying Digraphs

Let us first reformulate the original distributed min-max optimization problem (4.1). By introducing an auxiliary scalar variable $t \in \mathbb{R}$, it is straightforward to see that the problem (4.1) is equivalent to the following epigraph form,

$$\min_{\mathbf{x},t} \quad t \tag{4.7a}$$

$$\text{s. t.} \quad f_i(\mathbf{x}) \leq t, \ \forall i \in \mathcal{I}, \tag{4.7b}$$

$$\mathbf{x} \in \mathcal{X}. \tag{4.7c}$$

Note that the name "epigraph form" comes from the fact that the feasible set defined by (4.7b) is essentially the epigraph of the local cost function $f_i(\cdot)$. Particularly, let us denote the epigraph as the set $\mathcal{F}_i \subseteq \mathbb{R}^{p+1}$ where

$$\mathcal{F}_i := \Big\{(\mathbf{x}, t) \,|\, f_i(\mathbf{x}) \leq t\Big\}. \tag{4.8}$$

In addition, we bundle the two decision variables $\mathbf{x}$ and t as a new one $\boldsymbol{\theta} = [\mathbf{x}^\top, t]^\top \in \mathbb{R}^{p+1}$. On this basis, the epigraph form (4.7) of the min-max problem can be further reformulated as the following compact form

$$\min_{\boldsymbol{\theta}} \quad \mathbf{e}_{p+1}^\top \boldsymbol{\theta} \tag{4.9a}$$

$$\text{s. t.} \quad \theta \in (\cap_{i\in\mathcal{I}} \mathcal{F}_i) \cap (\mathcal{X} \times \mathbb{R}), \tag{4.9b}$$

where $\mathbf{e}_{p+1} \in \mathbb{R}^{p+1}$ is the $(p+1)$-th column of the identity matrix. Now, in order to solve the problem (4.9), we let each agent i maintain a local state $\boldsymbol{\theta}_i^k \in \mathbb{R}^{p+1}$ at each time-step $k \in \mathbb{N}_+$ and update it according to the following iterations,

$$\boldsymbol{\phi}_i^{k+1} = \sum\nolimits_{j \in \mathcal{N}_i^+} w_{ij}^k \boldsymbol{\theta}_j^k; \tag{4.10a}$$

$$\boldsymbol{\zeta}_i^{k+1} = \boldsymbol{\phi}_i^{k+1} - \alpha^k \cdot \mathbf{e}_{p+1}; \tag{4.10b}$$

$$\boldsymbol{\theta}_i^{k+1} = \Pi_{\mathcal{X}\times\mathbb{R}}\Big(\boldsymbol{\zeta}_i^{k+1} - \gamma^k \frac{\big[h_i(\boldsymbol{\zeta}_i^{k+1})\big]_+}{\|\nabla h_i(\boldsymbol{\zeta}_i^{k+1})\|^2} \nabla h_i(\boldsymbol{\zeta}_i^{k+1})\Big). \tag{4.10c}$$

In the above iteration, the function $h_i(\cdot) : \mathbb{R}^{p+1} \to \mathbb{R}$ is defined as $h_i(\boldsymbol{\theta}) := f_i(\mathbf{x}) - t$; w_{ij}^k's correspond to the non-negative weights when combining the information from the neighbors; $\alpha^k > 0$ and $\gamma^k > 0$ are two types of step-sizes; the operator $[\,\cdot\,]_+ : \mathbb{R} \to \mathbb{R}_+$ is defined as $[x]_+ := \max\{x, 0\}$; and $\Pi_{\mathcal{X}\times\mathbb{R}}(\cdot) : \mathbb{R}^{p+1} \to \mathcal{X} \times \mathbb{R}$ denotes the Euclidean projection on the set $\mathcal{X} \times \mathbb{R}$. Note that the gradient ∇h_i always has $\|\nabla h_i(\boldsymbol{\theta})\| \geq 1$, $\forall \boldsymbol{\theta} \in \mathbb{R}^{p+1}$, therefore the division in step (4.10b) is well-defined.

Remark 4.3.1. *The above iteration (4.10) can be also interpreted from the perspective of the classical distributed projected gradient descent. In fact, while noting that $\mathbf{e}_{p+1}$ is the gradient of the objective function in (4.9a), the step (4.10a) together with (4.10b) exactly perform the distributed gradient descent. In addition, to further cope with the constraints in (4.9b), the step (4.10c) takes two phases of projections: i) use the Polyak's projection to move the point $\boldsymbol{\zeta}_i^{k+1}$ towards the epigraph $\mathcal{F}_i$; and ii) use the Euclidean projection $\Pi_{\mathcal{X}\times\mathbb{R}}(\cdot)$ to ensure that the obtained point $\boldsymbol{\theta}_i^{k+1}$ is in the set $\mathcal{X} \times \mathbb{R}$. As already highlighted by the authors in [101] (see Remark 2 in [101]), one can also adopt the Euclidean projection to deal with the constraint induced by the epigraph $\mathcal{F}_i$, and thus reduce the iteration (4.10) into the standard distributed projected gradient descent algorithm. However, compared to the Euclidean projection, the Polyak's projection involves less computational cost at each iteration of the algorithm.*

Under some of the assumptions introduced in Chapter 1, the convergence result of the algorithm (4.10) is stated as the following theorem.

Theorem 4.3.1. *Under Assumptions 1.4.2, 1.4.3 and 1.4.4, suppose that the step-sizes α^k and γ^k are satisfied with i) $\sum_k^\infty \alpha^k = \infty$; ii) $\sum_k^\infty (\alpha^k)^2 < \infty$; and iii) $0 < \gamma^k < 2$, and let the time-varying weight matrix $W^k = [w_{ij}^k]_{i,j=1}^I$ be row-stochastic; see Assumption 1.4.10 and the underlying time-varying directed graph satisfy Assumption 1.4.6, then the sequence $\{\boldsymbol{\theta}_i^k\}_{k\in\mathbb{N}_+}$ generated by (4.10) converges to the exact optimal solution $\boldsymbol{\theta}^\star = [\mathbf{x}^{\star\top}, t^\star]^\top \in \mathbb{R}^{p+1}$ where $\mathbf{x}^\star \in \arg\min_{\mathbf{x}\in\mathcal{X}} G(\mathbf{x})$ and $t^\star = \max_{i\in\mathcal{I}} f_i(\mathbf{x}^\star)$, i.e.,*

$$\lim_{k\to\infty} \|\boldsymbol{\theta}_i^k - \boldsymbol{\theta}^\star\| = 0, \quad \forall \mathrm{i} \in \mathcal{I}. \tag{4.11}$$

Proof. The proof can be completed by following the similar path to that of Theorem 1 in [101], while ignoring the stochasticity in the considered problem. Therefore, we here omit the details. □

We shall highlight that, besides the above Assumptions stated in Theorem 4.3.1, another implicit condition is also crucial to ensure the convergence result; that is the perfection of the communications among all agents within the network. However, as shown in the two previous examples, the desired convergence can be easily disrupted by the potential network attacks. In order to fix such an issue, we next further enhance the algorithm (4.10) by integrating the resilient convex combination operator $\mathcal{R}(\cdot)$.

4.3.2 Integration with the Resilient Convex Combination

Let us recall that, during the iteration (4.10), the only process involved with communications is at the step (4.10a), in which each agent needs to combine (via weighted averaging) the information received from all neighbors. Therefore, to deal with the potential communication disruptions, a natural idea here is to replace the pure weighted averaging in (4.10a) by the resilient convex combination operator $\mathcal{R}(\cdot)$ introduced in Section 4.2. As such, we present our resilient solution method as the following Algorithm 5 for solving the distributed min-max optimization problem.

To ensure the convergence of Algorithm 5, a crucial part here is to guarantee that the conditions stated in Theorem 4.3.1 still hold while the weighted averaging has been re-

placed by the resilient convex combination. More precisely, one needs to make sure that the underlying time-varying directed graphs induced by $\mathcal{R}(\cdot)$ are satisfied with the B-strong connectivity condition; see Assumption 1.4.6. Towards this end, we will need the following result regarding the operator $\mathcal{R}(\cdot)$.

Algorithm 5: Resilient Distributed Min-Max Optimization Algorithm

Data: Each agent requires the number of the neighbors d_i and the upper bound of the number of attacked neighbors a_i^k. Set the step-sizes α^k and γ^k, and let $k = 0$.

while the termination criteria is NOT satisfied **do**

Each agent $i \in \mathcal{I}$ **simultaneously** does

`(S.1)` Receive the states $\boldsymbol{\theta}_j^k$ from all attacked and non-attacked neighbors, perform the resilient convex combination, and obtain

$$\boldsymbol{\phi}_i^{k+1} = \mathcal{R}_i(\boldsymbol{\theta}_{j\in\mathcal{N}_i^+}^k); \tag{4.12}$$

`(S.2)` Carry out the gradient descent step (4.10b), and obtain the intermediate result $\boldsymbol{\zeta}_i^{k+1}$;

`(S.3)` Update the state $\boldsymbol{\theta}_i^{k+1}$ via (4.10c), based on the obtained $\boldsymbol{\zeta}_i^{k+1}$;

`(S.4)` Let $k \to k+1$, and continue.

end

Proposition 4.3.1. *Suppose that the upper bound of the number of attacked neighbors has* $a_i^k \leq \lfloor (d_i+1)/(p+2) \rfloor - 1$ *where* $\lfloor \cdot \rfloor$ *is the floor function, then the optimal solution of the linear program* (4.6) *within the operator* $\mathcal{R}(\cdot)$ *must have* $\kappa^\star > 0$.

Proof. This proof is primarily based on the Reay's relaxed Tverberg conjecture [111], which is stated as follows: a set of m points in general position in the Euclidean space $\mathbb{R}^n$ is (r,k)-divisible if $m \geq (n+1)(r-1)+k+1$. Note that the (r,k)-divisibility here means the set of points can be partitioned into r disjoint subsets such that the intersection of the convex hulls of these r subsets is at least k-dimension. In this proof, we focus on the special case where $k = n$, and in this case, the Reay's relaxed Tverberg conjecture has been successfully proved when $2 \leq n \leq 8$ [112].

Now, let us denote $m = d_i + 1$, $n = p + 1$ and $r = a_i^k + 1$. Based on the above Reay's conjecture, if the condition in Proposition 4.3.1, i.e., $d_i + 1 \geq (p + 2)(a_i^k + 1)$, is satisfied, then the set of $d_i + 1$ points $\{\boldsymbol{\theta}_j^k\}_{j \in \mathcal{N}_i^+}$ in the space $\mathbb{R}^{p+1}$ is $(a_i^k + 1, p + 1)$-divisible. In other words, there exists a partition of $a_i^k + 1$ subsets, denoted as $\mathcal{P}_r,\ r = 1, 2, \cdots, a_i^k + 1$ where $\cup_{r=1}^{a_i^k+1} \mathcal{P}_r = \{\boldsymbol{\theta}_j^k\}_{j \in \mathcal{N}_i^+}$ and $\mathcal{P}_r \cap \mathcal{P}_{r'} = \emptyset$ when $r \neq r'$, such that the intersection set $\mathcal{T}$ of the corresponding convex hulls is $(p + 1)$-dimensional, i.e.,

$$\mathcal{T} := \bigcap_{r=1}^{a_i^k+1} \mathcal{H}(\mathcal{P}_r). \tag{4.13}$$

Next, we show that another intersection of convex hulls $\mathcal{C}_i^k$ which is generated by our resilient convex combination operator $\mathcal{R}(\cdot)$ satisfies with the condition $\mathcal{T} \subseteq \mathcal{C}_i^k$, and thus is also $(p + 1)$-dimensional when $d_i + 1 \geq (p + 2)(a_i^k + 1)$. Let us recall that the set $\mathcal{C}_i^k$ is expressed as

$$\mathcal{C}_i^k = \bigcap_{s=1}^{S_i^k} \mathcal{H}\Big(\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k\Big), \tag{4.14}$$

where each $\mathcal{Q}_i^k(s)$ is a subset of $\{\boldsymbol{\theta}_j^k\}_{j \in \mathcal{N}_i}$ satisfying that $|\mathcal{Q}_i^k(s)| = d_i - a_i^k$ with $s = 1, 2, \cdots, S_i^k$. To prove $\mathcal{T} \subseteq \mathcal{C}_i^k$, let us first show that, for each of the subsets $\mathcal{Q}_i^k(s)$, one of $\mathcal{P}_r$'s in the partition must be its subset, i.e., for $\forall s = 1, 2, \cdots, S_i^k$ there exists $1 \leq r_s \leq a_i^k + 1$ such that $\mathcal{P}_{r_s} \subseteq \mathcal{Q}_i^k(s)$. Now, we prove it by contradiction. Suppose that one can have an index $1 \leq \bar{s} \leq S_i^k$ such that $\mathcal{P}_r \nsubseteq \mathcal{Q}_i^k(\bar{s})$ for $\forall r = 1, 2, \cdots, a_i^k + 1$. Then, there exists at least one element in each subset $\mathcal{P}_r$ which is not contained in the set $\mathcal{Q}_i^k(\bar{s})$. Further, consider that all $\mathcal{P}_r$'s constitute the partition of $\{\boldsymbol{\theta}_j^k\}_{j \in \mathcal{N}_i}$, thus there are at least $a_i^k + 1$ elements in the set $\{\boldsymbol{\theta}_j^k\}_{j \in \mathcal{N}_i}$ which are not contained in $\mathcal{Q}_i^k(\bar{s})$. Consequently, the number of elements in $\mathcal{Q}_i^k(\bar{s})$ must have $|\mathcal{Q}_i^k(\bar{s})| \leq d_i - a_i^k - 1$, which contradicts the definition of the set $\mathcal{Q}_i^k(\bar{s})$. Taking advantages of the above conclusion, it follows that $\forall s = 1, 2, \cdots, S_i^k$,

$$\bigcap_{r=1}^{a_i^k+1} \mathcal{H}(\mathcal{P}_r) \subseteq \mathcal{H}(\mathcal{P}_{r_s}) \subseteq \mathcal{H}\Big(\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k\Big). \tag{4.15}$$

Therefore, we can have

$$\mathcal{T} = \bigcap_{r=1}^{a_i^k+1} \mathcal{H}(\mathcal{P}_r) \subseteq \bigcap_{s=1}^{S_i^k} \mathcal{H}\Big(\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k\Big) = \mathcal{C}_i^k. \tag{4.16}$$

Now, based on the fact that $\mathcal{C}_i^k$ is $(p+1)$-dimensional, it can be shown that any point $\widetilde{\boldsymbol{\theta}}$ in the interior part of $\mathcal{C}_i^k$, i.e., $\widetilde{\boldsymbol{\theta}} \in \text{int}(\mathcal{C}_i^k)$, can be expressed as

$$\widetilde{\boldsymbol{\theta}} = \sum_{j \in \mathcal{I}_i^k(s)} \beta_{ij}^k(s) \boldsymbol{\theta}_j^k, \tag{4.17}$$

where the index set $\mathcal{I}_i^k(s)$ must have $i \in \mathcal{I}_i^k(s) \subset \mathcal{N}_i^+$ and $|\mathcal{I}_i^k(s)| = d_i - a_i^k + 1$; and the weights $\beta_{ij}^k(s)$'s must have $\sum_{j \in \mathcal{I}_i^k(s)} \beta_{ij}^k(s) = 1$ and $\beta_{ij}^k(s) > 0, \forall s = 1, 2, \cdots, S_i^k$. In fact, for each $s = 1, 2, \cdots, S_i^k$, let us denote the pure average of the points in $\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k$ as

$$\overline{\boldsymbol{\theta}} := \frac{1}{|\mathcal{I}_i^k(s)|} \cdot \sum_{j \in \mathcal{I}_i^k(s)} \boldsymbol{\theta}_j^k. \tag{4.18}$$

Consider that the point $\widetilde{\boldsymbol{\theta}}$ is in the interior part of $\mathcal{C}_i^k$ and thus is also in the interior part of the set $\mathcal{H}\Big(\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k\Big)$, then there must exist a small constant $\epsilon > 0$ such that

$$\widehat{\boldsymbol{\theta}} := (1+\epsilon) \cdot \widetilde{\boldsymbol{\theta}} - \epsilon \cdot \overline{\boldsymbol{\theta}} \in \text{int}\Big(\mathcal{H}\Big(\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k\Big)\Big). \tag{4.19}$$

On this basis, we can represent the interior point $\widetilde{\boldsymbol{\theta}}$ as

$$\begin{aligned} \widetilde{\boldsymbol{\theta}} &= \frac{1}{1+\epsilon} \cdot \widehat{\boldsymbol{\theta}} + \frac{\epsilon}{1+\epsilon} \cdot \overline{\boldsymbol{\theta}} \\ &= \frac{1}{1+\epsilon} \cdot \widehat{\boldsymbol{\theta}} + \frac{\epsilon}{|\mathcal{I}_i^k(s)|(1+\epsilon)} \cdot \sum_{j \in \mathcal{I}_i^k(s)} \boldsymbol{\theta}_j^k. \end{aligned} \tag{4.20}$$

Note that $\widehat{\boldsymbol{\theta}}$ can be also expressed as the convex combination of the points in $\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k$, therefore (4.17) must be true with $\sum_{j \in \mathcal{I}_i^k(s)} \beta_{ij}^k(s) = 1$ and $\beta_{ij}^k(s) > 0, \forall s = 1, 2, \cdots, S_i^k$.

As a result of the above result, it is straightforward to verify that any interior point of the generated set $\mathcal{C}_i^k$ be expressed as the non-zero convex combination of the points in

$\mathcal{Q}_i^k(s) \cup \boldsymbol{\theta}_i^k$, $\forall s = 1, 2, \cdots, S_i^k$. Therefore, to maximize the minimum weight $\beta_{ij}^k(s)$ as shown in the linear program (4.6), the optimal solution must have $\kappa^\star > 0$. The proof is completed. □

Based on the above Proposition 4.3.1, a direct corollary can be stated as follows.

Corollary 4.3.1. *Suppose that $a_i^k \leq \lfloor (d_i + 1)/(p + 2) \rfloor - 1$ is satisfied with each agent i at each time-step k, then the output of the resilient convex combination can be equivalently expressed as*

$$\boldsymbol{\phi}_i^{k+1} = \sum_{j \in \mathcal{N}_i^+ \setminus \mathcal{A}_i^k} \mu_{ij}^k \boldsymbol{\theta}_j^k, \tag{4.21}$$

where the weights μ_{ij}^k's must have $\sum_{j \in \mathcal{N}_i^+ \setminus \mathcal{A}_i^k} \mu_{ij}^k = 1$ and $\mu_{ij}^k > 0$, $\forall j \in \mathcal{N}_i^+ \setminus \mathcal{A}_i^k$.

Proof. According to the procedure of the operator $\mathcal{R}(\cdot)$; see Algorithm 4, its output can be expressed as

$$\boldsymbol{\phi}_i^{k+1} = \sum_{j \in \mathcal{I}_i^k(s)} \beta_{ij}^k(s) \boldsymbol{\theta}_j^k, \tag{4.22}$$

where the index set $\mathcal{I}_i^k(s)$ must have $i \in \mathcal{I}_i^k(s) \subset \mathcal{N}_i^+$ and $|\mathcal{I}_i^k(s)| = d_i - a_i^k + 1$. Clearly, there are totally $S_i^k = \binom{d_i}{d_i - a_i^k}$ possibilities of the set $\mathcal{I}_i^k(s)$, therefore we let $s = 1, 2, \cdots, S_i^k$ to cover all the possibilities. Based on the result in Proposition 4.3.1, the optimal solution $\kappa^\star > 0$ simply implies that

$$\beta_{ij}^k(s) > 0, \ \forall j \in \mathcal{I}_i^k(s) \text{ and } s = 1, 2, \cdots, S_i^k. \tag{4.23}$$

Further, we consider some special cases of the possibility s in which the set $\mathcal{I}_i^k(s)$ only contains the indices of non-attacked neighbors, i.e., $\mathcal{I}_i^k(s) \cap \mathcal{A}_i^k = \emptyset$. Note that, since the number of attacked neighbors is upper bounded by a_i^k, such special cases of s must exist. In addition, let us denote the set of all special cases as $\chi_i^k \subset \{1, 2, \cdots S_i^k\}$, then it holds that

$$\bigcup_{s \in \chi_i^k} \mathcal{I}_i^k(s) = \mathcal{N}_i^+ \setminus \mathcal{A}_i^k. \tag{4.24}$$

Notice that the expression (4.22) holds for $\forall s = 1, 2, \cdots, S_i^k$ and thus for $\forall s \in \chi_i^k$. Therefore, the output of the operator $\mathcal{R}(\cdot)$ can be further represented as

$$\boldsymbol{\phi}_i^{k+1} = \frac{1}{|\chi_i^k|} \cdot \sum_{s \in \chi_i^k} \sum_{j \in \mathcal{I}_i^k(s)} \beta_{ij}^k(s) \boldsymbol{\theta}_j^k = \sum_{j \in \mathcal{N}_i^+ \setminus \mathcal{A}_i^k} \mu_{ij}^k \boldsymbol{\theta}_j^k, \tag{4.25}$$

where $\mu_{ij}^k = 1/|\chi_i^k| \cdot \sum_{s \in \chi_{i,j}^k} \beta_{ij}^k(s)$ with the set $\chi_{i,j}^k$ being

$$\chi_{i,j}^k := \{s \,|\, s \in \chi_i^k,\ j \in \mathcal{I}_i^k(s)\}. \tag{4.26}$$

Now, by (4.24), it is easy to see that $\forall j \in \mathcal{N}_i^+ \setminus \mathcal{A}_i^k$, $\chi_{i,j}^k \neq \emptyset$, and thus $\mu_{ij}^k > 0$. The proof is completed. □

We shall emphasize that the above Corollary 4.3.1 essentially ensures three key properties of the step `(S.2)` in Algorithm 5: i) without identifying the attacked neighbors, each agent i can achieve the result $\boldsymbol{\phi}_i^{k+1}$ which integrates the information from only the non-attacked neighbors; ii) by ensuring $\mu_{ij}^k > 0$, it is guaranteed that the agent truly integrates the information from the neighbors; and iii) the row-stochasticity condition of the underlying weight matrices is also satisfied by $\sum_{j \in \mathcal{N}_i^+ \setminus \mathcal{A}_i^k} \mu_{ij}^k = 1$. Taking advantages of Corollary 4.3.1 together with the previous assumptions, we are now ready to present the convergence of our resilient distributed min-max algorithm.

Theorem 4.3.2. *Under Assumptions 1.4.2, 1.4.3 and 1.4.4 as well as the conditions in Corollary 4.3.1, suppose that the step-sizes α^k and γ^k have i) $\sum_k^\infty \alpha^k = \infty$; ii) $\sum_k^\infty (\alpha^k)^2 < \infty$; and iii) $0 < \gamma^k < 2$, and let the original time-invariant directed graph $\mathcal{G}$ be satisfied with Assumption 1.4.7, then the sequence $\{\boldsymbol{\theta}_i^k\}_{k \in \mathbb{N}_+}$ generated by Algorithm 5 converges to the exact optimal solution $\boldsymbol{\theta}^\star$, i.e.,*

$$\lim_{k \to \infty} \|\boldsymbol{\theta}_i^k - \boldsymbol{\theta}^\star\| = 0, \quad \forall i \in \mathcal{I}. \tag{4.27}$$

Proof. Let us represent the underlying weight matrices as $M^k := [\mu_{ij}^k]_{i,j=1}^N$, which is induced by the resilient convex combination operator $\mathcal{R}(\cdot)$; see (4.21). Note that we set $\mu_{ij}^k = 0$ if $j \notin \mathcal{N}_i^+ \setminus \mathcal{A}_i^k$. Then, according to Corollary 4.3.1, it is ensured that M^k is row-stochastic for

$\forall k \in \mathbb{N}_+$. In addition, if we denote $\bar{\mathcal{G}}^k = (\mathcal{N}, \bar{\mathcal{E}}^k)$ the time-varying digraph which is defined by the weight matrix M^k, then the set of directed edges $\bar{\mathcal{E}}^k$ can be represented as

$$\bar{\mathcal{E}}^k = \Big\{(i,j) \,|\, (i,j) \in \mathcal{E}, j \notin \mathcal{A}^k, i,j \in \mathcal{I}\Big\}, \tag{4.28}$$

where $\mathcal{E}$ is the set of edges of the original time-invariant, directed graph $\mathcal{G}$.

Now, based on Assumption 4.1.1, there exists an integer $T > 0$ such that $\cap_{t=k}^{T+k-1} \mathcal{A}^t = \emptyset$, therefore, it is straightforward to see that $\cup_{t=k}^{T+k-1} \mathcal{E}^t = \mathcal{E}$. Since it is assumed in Assumption 1.4.7 that the original digraph $\mathcal{G}$ is strongly-connected, hence the time-varying digraphs $\bar{\mathcal{G}}^k$ must be T-strongly connected, i.e., the joint digraph $\cup_{t=k}^{T+k-1} \bar{\mathcal{G}}^t = (\mathcal{N}, \cup_{t=k}^{T+k-1} \bar{\mathcal{E}}^t)$ is strongly-connected.

As a consequence, we can conclude that the assumption in Theorem 4.3.1 holds with respect to the underlying time-varying digraphs $\bar{\mathcal{G}}^k$'s. Based on Theorem 4.3.1, the convergence result of Algorithm 5 is proved. □

5. A DISTRIBUTED PROGRESSIVE HEDGING METHOD FOR SOLVING TWO-STAGE STOCHASTIC PROGRAMS

This chapter adapts the well-known PH method under a peer-to-peer multi-agent network for solving two-stage stochastic programs efficiently. Similar to the existing parallel PH method, our distributed PH (DistPH) algorithm assigns each agent to take charge of the computing tasks covering one or few scenarios, and thus it distributes the overall computational burden of solving the stochastic program over the entire network. However, unlike the parallel PH method, the DistPH algorithm no longer needs a master node to realize central coordination, and thus it distributes its communications workload over the network as well. In this chapter, we prove the exact convergence of the DistPH algorithm for two-stage stochastic programs with continuous variables subject to convex constraints. In addition, we investigate several computational issues for the mixed-integer cases to improve the adaptation efficiency.

5.1 Problem Statement

Two-stage (or multi-stage) stochastic programming provides a promising framework for solving real-world planning problems where decisions are made in stages under uncertainty. A standard formulation of two-stage stochastic program reads

$$\min_{\mathbf{x}\in\mathcal{X}} \quad \mathbf{c}^\top \mathbf{x} + Q(\mathbf{x}), \tag{5.1}$$

where the function $Q(\mathbf{x})$ is typically defined as the following expectation form,

$$Q(\mathbf{x}) := \mathbb{E}_\xi \Big[\min_{\mathbf{y}\in\mathcal{Y}_\xi(\mathbf{x})} \{\mathbf{q}_\xi^\top \mathbf{y}\} \Big]. \tag{5.2}$$

In the formulation, $\mathbf{x} \in \mathbb{R}^{p_1}$ and $\mathbf{y} \in \mathbb{R}^{p_2}$ denote the first-stage and second-stage decision variables respectively, and $\mathbf{c} \in \mathbb{R}^{p_1}$ is a fixed and known parameter vector corresponding to the first stage. The uncertainty, coming into the second stage, is represented by a random variable $\boldsymbol{\xi}$. The vector $\mathbf{q}_\xi \in \mathbb{R}^{p_2}$ is an uncertain parameter vector dependent on $\boldsymbol{\xi}$, and function $Q(\cdot) : \mathbb{R}^{p_2} \to \mathbb{R}$ computes the expectation of the second-stage minimizer with respect

to $\boldsymbol{\xi}$. In the formulation, the constraints, defined by feasible sets $\mathcal{X} \subseteq \mathbb{R}^{p_1}$ and $\mathcal{Y}_\xi(\mathbf{x}) \subseteq \mathbb{R}^{p_2}$, are typically composed of a group of linear equalities and/or inequalities, possibly with some integer or binary restrictions. Note that the problem is known as stochastic mixed-integer program (SMIP), when integerality restriction appears in the first and/or second stage.

To deal with the two-stage stochastic program (5.1)–(5.2), a widely used approach builds on the idea of representing the uncertainty with a finite number of scenarios. That is, the random variable $\boldsymbol{\xi}$ is assumed to be taken from a discrete distribution with finite support, i.e., $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \cdots, \boldsymbol{\xi}_I\}$, where associated with each $\boldsymbol{\xi}_i, i \in \mathcal{I} = \{1, 2, \cdots, I\}$, is the scenario-based parameter $\mathbf{q}_i \in \mathbb{R}^{p_2}$, feasible set $\mathcal{Y}_i(\mathbf{x}) \subseteq \mathbb{R}^{p_2}$, and corresponding probability p_i satisfying with $\sum_{i=1}^{I} p_i = 1$. By doing so, the original problem (5.1)–(5.2) can be equivalently reformulated as the following extensive form,

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{y}_i} \quad & \mathbf{c}^\top \mathbf{x} + \sum_{i=1}^{I} p_i \mathbf{q}_i^\top \mathbf{y}_i, \\ \text{s. t.} \quad & (\mathbf{x}, \mathbf{y}_i) \in \mathcal{K}_i, \ \forall i \in \mathcal{I}, \end{aligned} \tag{5.3}$$

where we combine the feasible sets $\mathcal{X}$ and $\mathcal{Y}_i$ for the two stages, and denote it with a compact form $\mathcal{K}_i$, i.e. $\mathcal{K}_i := \Big\{(\mathbf{x}, \mathbf{y}_i) | \mathbf{x} \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}_i(\mathbf{x})\Big\}$.

For scenario-based reformulation (5.3), the PH method is commonly considered as an efficient solution method. It is proved to achieve exact convergence when the first- and second-stage decision variables in (3) are both continuous and subject to convex constraints [78]. Although there is no theoretical guarantee for the convergence when it comes to the mixed-integer cases, yet the PH method has been successfully applied as a heuristic approach which can find high-quality solutions for solving SMIPs [113]. Several key issues, such as the choice of penalty parameter and termination criterion, are investigated in [114] when applying the PH method to the mixed-integer cases. Moreover, it is shown in [115] that a lower bound can be computed in any iteration of the PH method, which allows to assess the quality of iterative solutions to a SMIP.

Another benefit of the PH method is often attributed to its straightforward parallelization [83], [84]. That is, the PH method allows to deal with mutually independent scenario-specific subproblems separately; see details in Section 5.2, and a number of processors can

be utilized to execute the algorithm in parallel. As a result, the computational cost is spread over all processors and thus reduced for each single processor. Further, to realize such parallelization, a master node is typically designated to provide such coordination, and a group of worker nodes are assigned to solve separate subproblems. According to this configuration, the PySP package [116], as a robust off-the-shelf solver, has implemented the parallel execution of the PH method. Significant parallel efficiency can be achieved, especially when the scenario-specific subproblems are difficult to solve [117]. Parallel computing has been broadly investigated for many other approaches to solving stochastic programs. For example, a parallel sub-gradient method for computing Lagrangian duals is presented in [118], together with its asynchronous variant. Another parallel sub-gradient based method is proposed in [119], particularly for solving a stochastic unit commitment problem. A parallel cutting-plane method is studied in [120], which utilizes the sub-gradient information to build cutting planes at each iteration.

It is worthy noting that all these methods are developed by assuming a master-worker architecture. To address the computational challenges arising in a master-worker architecture and follow the recent attraction to peer computing algorithm design, we adapt the parallel PH method under a peer-to-peer multi-agent computing network, where there is no presence of a master node. Our motivations are two-fold. First, under a master-worker architecture, since the master node is expected to provide coordination among all worker nodes, the communication burden on it, e.g., communication bandwidth required, could be extremely heavy. It is even prohibitive in many practical applications to build such a powerful master node that can maintain communication channels with all other nodes within the network. Second, with the existence of the master node, cyber-security issue may also arise as any failure on it can lead to the collapse of entire master-worker architecture. Therefore, we aim to develop our distributed solution method under a peer-to-peer network, which solely consists of peer computing agents (nodes) and each of them is directly connected with a subset of nodes as its neighbors.

By enabling each agent to solve the corresponding scenario-specific subproblem and exchanging information with its neighbors, we introduce a distributed update scheme for the PH method which merely relies on peer communications between neighboring agents. It is

emphasized that, in such a distributed framework, no agent serves as the master node but each one plays an equal role in contributing to the computation. As a consequence, we expect that aforementioned communication burden for the master node is spread over the network and separate communications can be executed simultaneously among multiple peer agents. Furthermore, considering that the distributed framework is more compatible with general connected networks, it would be more flexible and applicable in real-world applications. We should also highlight that the distributed framework exhibits more robustness against the potential security attacks, as it can be still functional even if one or few agents fail. Note that such a peer-to-peer computing network has been widely adopted in solving deterministic optimization problems; see the detailed literature review in Chapter 1. However, to the best of our knowledge, we are not aware of any work in the existing literature that utilizes a peer-to-peer network to solve (two-stage) stochastic programs. Finally, we remark that the master-worker architecture can be regarded as a special case of the peer-to-peer network, since the central node can serve as a master when the considered network has a star topology.

5.2 The Basic PH Method

In this section, we introduce the basic PH method [78], and see how far it is from being adapted under a peer-to-peer computing network. Recall that our goal is to solve the scenario-based formulation (5.3). Such a formulation has a special structure that can be exploited algorithmically by decomposition methods [79], [120]. Suppose that there are I scenarios in total. To leverage the decomposable structure, the first-stage decision variable $\mathbf{x}$ is split into I scenario-dependent copies, i.e., $\mathbf{x}_i \in \mathbb{R}^{p_1}$, $i \in \mathcal{I}$. As a result, formulation (5.3) can be further rewritten as

$$\begin{aligned} \min_{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}} \quad & \sum_{i=1}^{I} p_i (\mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i), \\ \text{s. t.} \quad & \mathbf{x}_i = \mathbf{z}, \ (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{K}_i, \ \forall i \in \mathcal{I}. \end{aligned} \tag{5.4}$$

Note that the newly introduced variable $\mathbf{z} \in \mathbb{R}^{p_1}$ and constraints $\mathbf{x}_i = \mathbf{z}$ are intended to ensure the so-called non-anticipativity. That is, to require $\mathbf{x}_i$ for each individual scenario i to achieve an agreement on $\mathbf{z}$ eventually.

For an optimization problem with such non-anticipativity constraints, the basic PH method is commonly regarded as a viable solution method, which can be represented by the following three main steps:

`primal-update-step`:

$$(\mathbf{x}_i^{k+1}, \mathbf{y}_i^{k+1}) = \underset{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{K}_i}{\arg\min}\, L_i^{\rho}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}^k, \boldsymbol{\omega}_i^k); \tag{5.5}$$

`aggregation-step`:

$$\mathbf{z}^{k+1} = \sum_{s=1}^{I} p_i \mathbf{x}_i^{k+1}; \tag{5.6}$$

`dual-update-step`:

$$\boldsymbol{\omega}_i^{k+1} = \boldsymbol{\omega}_i^k + \rho(\mathbf{x}_i^{k+1} - \mathbf{z}^{k+1}). \tag{5.7}$$

Here, each $\boldsymbol{\omega}_i \in \mathbb{R}^{p_1}$ serves as a dual variable, ρ is a penalty parameter, and each function $L_i^{\rho} : \mathbb{R}^{p_1} \times \mathbb{R}^{p_2} \times \mathbb{R}^{p_1} \times \mathbb{R}^{p_1} \to \mathbb{R}$ is defined as

$$L_i^{\rho}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}, \boldsymbol{\omega}_i) := \mathbf{c}^{\top}\mathbf{x}_i + \mathbf{q}_i^{\top}\mathbf{y}_i + \boldsymbol{\omega}_i^{\top}(\mathbf{x}_i - \mathbf{z}) + \frac{\rho}{2}\|\mathbf{x}_i - \mathbf{z}\|^2. \tag{5.8}$$

It has been shown in [78] that, when decision variables for both stages are continuous and subject to convex constraints, performing the above three basic PH steps (5.5)–(5.7) iteratively can lead each $\mathbf{x}_i^k$ to an agreement on a common first-stage optimal solution $\mathbf{x}^{\star}$, i.e., $\mathbf{x}_i^k \to \mathbf{x}^{\star}$, $\forall i \in \mathcal{I}$, and also drive $\mathbf{y}_i^k$ to the second-stage optimal solution $\mathbf{y}_i^{\star}$ for each scenario i, i.e., $\mathbf{y}_i^k \to \mathbf{y}_i^{\star}$, $\forall i \in \mathcal{I}$.

Next, to adapt steps (5.5)–(5.7) under a general peer-to-peer computing network, we suppose that each agent is corresponding to one single scenario i and only takes charge of its own primal and dual variables, i.e., $(\mathbf{x}_i^k, \mathbf{y}_i^k)$ and $\boldsymbol{\omega}_i^k$. It is still expected that the same convergence, i.e., $\mathbf{x}_i^k \to \mathbf{x}^{\star}$ and $\mathbf{y}_i^k \to \mathbf{y}_i^{\star}$, $\forall i \in \mathcal{I}$, can be achieved, while subject to peer-to-peer network topology. Considering that only peer communications are allowed in the desired

distributed framework, thus each agent can merely access the information from itself and its immediate neighbors. Keeping this in mind, it is worth noting that the primal and dual update steps (5.5) and (5.7) are readily implementable under the peer-to-peer network, as they simply involve the information maintained by each agent itself. However, implementing the aggregation step (5.6) requires each agent to access the information from all other agents within the network. Thus, it is the aggregation step that prevents the basic PH method from being adapted under a distributed framework. This is also the reason why a master-worker architecture is typically assumed for running the parallel PH method at present. In the next section, we will address the issue of the aggregation step, and further design the distributed variant of the PH method.

5.3 The DistPH Method

5.3.1 Basic Algorithm Design

As just pointed out in the preceding analysis, the barrier that prevents the basic PH method from being implementable in a distributed manner is the aggregation step. In details, we remark that variable $\mathbf{z}$ in (5.4) serves as a commonly shared variable, requesting all agents to achieve an agreement on the first-stage decisions, i.e., non-anticipativity constraint. With non-anticipativity constraint, updating $\mathbf{z}$ requires information from all agents, as shown in the aggregation step (5.6). In response, we next eliminate this requirement by establishing a relationship only between neighboring agents.

Instead of using a commonly shared $\mathbf{z}$ variable for all agents, we use its multiple copies $\mathbf{z}_{ij}$'s with each $\mathbf{z}_{ij} \in \mathbb{R}^{p_1}$ corresponding to a neighboring agents pair $(i,j) \in \mathcal{E}$. Consequently, updating the new variable $\mathbf{z}_{ij}$ will only involve the communication between agents i and j. Thus, formulation (5.4) is further rewritten as the following equivalent form,

$$\begin{aligned} \min_{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_{ij}} \quad & \sum_{i=1}^{I} p_i(\mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i), \\ \text{s. t.} \quad & \mathbf{x}_i = \mathbf{z}_{ij},\ \mathbf{x}_j = \mathbf{z}_{ij},\ \forall (i,j) \in \mathcal{E}, \\ & (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{K}_i,\ \forall i \in \mathcal{I}. \end{aligned} \tag{5.9}$$

Note that the equivalence between (5.9) and (5.4) relies the connectedness of the underlying time-invariant undirected graph; see Assumption 1.4.8 in Chapter 1

Let us further rewrite formulation (5.9) with a more compact form. Recall that $\mathbf{x} \in \mathbb{R}^{Ip_1}$ denotes the concatenated variable which stack individual $\mathbf{x}_i$'s as defined in the previous chapters, and similarly, we let $\mathbf{y} := [\mathbf{y}_1^\top, \mathbf{y}_2^\top, \cdots, \mathbf{y}_I^\top]^\top \in \mathbb{R}^{Ip_2}$. Subsequently, the overall objective function is expressed as

$$F(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^{I} p_i(\mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i), \tag{5.10}$$

and formulation (5.9) is equivalent to,

$$\begin{aligned} \min_{\mathbf{x},\,\mathbf{y}} \quad & F(\mathbf{x}, \mathbf{y}), \\ \text{s. t.} \quad & (\mathbf{x}, \mathbf{y}) \in \mathcal{K},\ \mathbf{L}\mathbf{x} = \mathbf{0}, \end{aligned} \tag{5.11}$$

where $\mathbf{L} := L \otimes \mathbf{I}_{p_1}$ and L denotes the Laplacian matrix of the time-invariant undirected graph. We note that independent constraints $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{K}_i,\ \forall i \in \mathcal{I}$ in (5.9) have been combined as $(\mathbf{x}, \mathbf{y}) \in \mathcal{K}$ in (5.11), with $\mathcal{K} := \{(\mathbf{x}, \mathbf{y}) \,|\, (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{K}_i)\}$. In addition, given the connectedness of the graph $\mathcal{G}$ by Assumption 1.4.8, the null space of the Laplacian matrix L is spanned by the one vector $\mathbf{1}_I$ and thus the constraint $\mathbf{L}\mathbf{x} = \mathbf{0}$ ensures the non-anticipativity solutions.

Now that formulation (5.11) is in a form consistent with the one investigated in [78], we update the compact variables with the following steps,

`primal-update-step`:

$$(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) = \underset{(\mathbf{x}, \mathbf{y}) \in \mathcal{K}}{\arg\min} \quad F(\mathbf{x}, \mathbf{y}) + \mathbf{x}^\top \boldsymbol{\nu}^k + \frac{\rho}{2} \|\mathbf{x} - \hat{\mathbf{z}}^k\|^2; \tag{5.12}$$

`aggregation-step`:

$$\hat{\mathbf{z}}^{k+1} = J\mathbf{x}^{k+1}; \tag{5.13}$$

`dual-update-step`:

$$\boldsymbol{\nu}^{k+1} = \boldsymbol{\nu}^k + \rho K \mathbf{x}^{k+1}. \tag{5.14}$$

Similarly, the dual variable $\boldsymbol{\nu}^k \in \mathbb{R}^{Ip_1}$ stacks all the individual $\boldsymbol{\nu}_i^k$'s, i.e., $\boldsymbol{\nu} = [\boldsymbol{\nu}_1^\top, \boldsymbol{\nu}_2^\top, \cdots, \boldsymbol{\nu}_I^\top]^\top$, and the two key operators $J : \mathbb{R}^{Ip_1} \to \mathbb{R}^{Ip_1}$ and $K : \mathbb{R}^{Ip_1} \to \mathbb{R}^{Ip_1}$ are defined as

$$J := \frac{\mathbf{I}_I + D^{-1}A}{2} \otimes \mathbf{I}_{p_1} \quad \text{and} \quad K := (\mathbf{I}_I - J) \otimes \mathbf{I}_{p_1} = \frac{\mathbf{I}_I - D^{-1}A}{2} \otimes \mathbf{I}_{p_1}. \tag{5.15}$$

Inherited from the convergence properties of the original PH method in [78], the above iterative updating scheme (5.12)–(5.14) achieves the same convergence result, as we state with the following proposition.

Proposition 5.3.1. *Suppose that the underlying time-invariant undirected graph $\mathcal{G}$ satisfies Assumption 1.4.8 and the feasible set $\mathcal{K}_i$ is closed and convex for all $i \in \mathcal{I}$, then the sequences $\{\mathbf{x}^k\}_{k=1}^\infty$ and $\{\mathbf{y}^k\}_{k=1}^\infty$ generated by the iterative procedure (5.12) – (5.14) with arbitrary initialization, have the following convergence,*

$$\mathbf{x}_i^k \to \mathbf{x}^\star \quad \textit{and} \quad \mathbf{y}_i^k \to \mathbf{y}_i^\star, \quad \forall i \in \mathcal{I}, \tag{5.16}$$

if the first-stage optimal solution $\mathbf{x}^\star$ and the second-stage optimal solutions $\mathbf{y}_i^\star$ exist for solving problem (5.11), or equivalently problem (5.4).

Proof. The proof of Proposition 5.3.1 follows that for Theorem 5.1 in [78], while the operator K needs to be adapted into the one defined in (5.15). Note that K reflects the network topology. Furthermore, it plays the same role as in [78], since the null space of K (identical to the null space of Laplacian matrix L) contains all the solutions that satisfy the non-anticipativity constraints. □

Proposition 5.3.1 above provides a theoretical guarantee for convergence of the updating scheme (5.12)–(5.14) when considering a convex case, i.e., only continuous decision variables are involved and subject to convex constraints. However, since the scheme is designed from a global perspective, it remains unknown if one can implement it in the desired distributed framework. Hence, we next verify, from the standpoint of each individual agent, that steps (5.12)–(5.14) are implementable in a distributed manner.

As earlier, we first specify that the variable $\hat{\mathbf{z}}^k$ is defined as in the compact form, i.e., $\hat{\mathbf{z}}^k = [(\hat{\mathbf{z}}_1^k)^\top, (\hat{\mathbf{z}}_2^k)^\top, \cdots, (\hat{\mathbf{z}}_I^k)^\top]^\top$. Then, by the aggregation step (5.13) and the definition of the operator J, updating each $\hat{\mathbf{z}}_i^k$ can be done by

$$\hat{\mathbf{z}}_i^k = \frac{1}{2}\mathbf{x}_i^k + \frac{1}{2d_i}\sum_{j\in\mathcal{N}_i}\mathbf{x}_j^k. \tag{5.17}$$

Note that the primal variable $\mathbf{x}_i^k$ is privately maintained by agent i and $\mathcal{N}_i$ denotes the set of its neighbors, thus step (5.17) is naturally implementable under a peer-to-peer multi-agent network. Now with the help of the aggregation variable $\hat{\mathbf{z}}_i^k$, the DistPH method performs the following steps,

`primal-update-step`:

$$(\mathbf{x}_i^{k+1}, \mathbf{y}_i^{k+1}) = \underset{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{K}_i}{\arg\min}\left\{ p_i(\mathbf{c}^\top\mathbf{x}_i + \mathbf{q}_i^\top\mathbf{y}_i) + (\boldsymbol{\nu}_i^k)^\top\mathbf{x}_i + \frac{\rho}{2}\|\mathbf{x}_i - \hat{\mathbf{z}}_i^k\|^2\right\}; \tag{5.18}$$

`aggregation-step`:

$$\hat{\mathbf{z}}_i^{k+1} = \frac{1}{2}\mathbf{x}_i^{k+1} + \frac{1}{2d_i}\sum_{j\in\mathcal{N}_i}\mathbf{x}_j^{k+1}; \tag{5.19}$$

`dual-update-step`:

$$\boldsymbol{\nu}_i^{k+1} = \boldsymbol{\nu}_i^k + \rho\Big(\mathbf{x}_i^{k+1} - \hat{\mathbf{z}}_i^{k+1}\Big). \tag{5.20}$$

Algorithm 6: DistPH Method

Data: Each agent i specifies the penalty parameter ρ and initializes $\mathbf{x}_i^0$, $\mathbf{y}_i^0$ and $\boldsymbol{\nu}_i^0$. Let $k = 0$.

while a termination criterion is NOT satisfied **do**

Each agent $i \in \mathcal{I}$ **simultaneously** does

`(S.1)` Updates the primal variables $(\mathbf{x}_i^{k+1}, \mathbf{y}_i^{k+1})$ by step (5.18), and broadcasts the updated $\mathbf{x}_i^{k+1}$ to its neighbors;

`(S.2)` Receives $\mathbf{x}_j^{k+1}$ from its neighbors, and performs the step (5.19);

`(S.3)` Updates the dual variable $\boldsymbol{\nu}_i^{k+1}$ by step (5.20);

`(S.4)` Let $k \leftarrow k+1$, and continue.

end

We note that, with the above updating scheme, each agent can simply receive information of previous iteration from all its neighbors, perform the steps (5.18)–(5.20) for itself, and broadcast the updated variables to its neighbors. By doing this, we have eliminated the requirement of the master node and adapted the PH method under a general peer-to-peer network. To summarize, we outline all steps of the DistPH method as Algorithm 6, and provide its convergence result as a corollary of the above Proposition 5.3.1.

Corollary 5.3.1. *Suppose that the conditions in Theorem 5.3.1 hold, then the sequences* $\{\mathbf{x}_i^k\}_{k=1}^{\infty}$ *and* $\{\mathbf{y}_i^k\}_{k=1}^{\infty}$ *generated by Algorithm 6 with arbitrary initialization, have the following convergence,*

$$\mathbf{x}_i^k \to \mathbf{x}^\star \quad \textit{and} \quad \mathbf{y}_i^k \to \mathbf{y}_i^\star, \quad \forall i \in \mathcal{I}. \tag{5.21}$$

Remark 5.3.1. *Unlike the standard parallel PH method in which the master node performs the aggregation step by simply computing the weighted average* $\mathbf{z}^k = \sum_{i=1}^{I} p_i \mathbf{x}_i^k$*, our distributed algorithm let each agent maintain its own aggregation variable* $\hat{\mathbf{z}}_i^k$ *and update it by itself according to (5.17). This updating scheme helps eliminate the requirement of the master node, thus enabling the adaptation of the PH method under a peer-to-peer computing network. It increases the overall communication cost within the entire network at each iteration of the algorithm, depending on the underlying topology. The denser the network is, the more communication resources are required to perform the distributed computing. However, we shall remark that, like the computation side, the overall communication cost is spread over network agents, and each agent can simultaneously execute its own portion of the communication tasks. In addition, knowing that the core of the PH method is to deal with the non-anticipativity constraints, in principle, when the underlying network has a denser topology, a smaller number of iterations would be demanded for our distributed method to produce non-anticipativity solutions. Subsequently, it can be seen that the investigation of communication cost should not only involve the network topology, but also consider the algorithm convergence rate, which complicates a comprehensive complexity analysis.*

Remark 5.3.2. *When the DistPH method is setup over a fully-connected network (i.e., with all-to-all communications), it works algorithmically in a nearly identical way with its parallel counterpart. However, one should note that there are slight differences in the choice of penalty parameter ρ and the computation of aggregation variables. That is, the DistPH method i) scales the parameter ρ by the probability p_i associated with each scenario i; and ii) computes the aggregation variable $\hat{\mathbf{z}}_i^k = \mathbf{x}_i^k/2 + 1/2 \cdot \sum_{j \neq i}^{I} \mathbf{x}_j^k/(I-1)$ by assigning more weight to the local variable $\mathbf{x}_i^k$, instead of taking the weighted average $\sum_{i=1}^{I} p_i \cdot \mathbf{x}_i^k$ relying on the probabilities p_i. Despite these differences, we note that both parallel and distributed PH methods work on the same principle, i.e., solving scenario-specific subproblems while taking non-anticipativity constraints into account. It seems that the DistPH method is more conservative when performing the aggregation step, as it assigns more weight to the local variable $\mathbf{x}_i^k$. As a result, the update $\mathbf{x}_i^{k+1}$ tends to be not far away from the incumbent. However, we still remark that it is unknown whether such conservatism will theoretically slow down or speed up the convergence.*

Remark 5.3.3. *It is well-known that the basic PH method has a close connection with the alternating direction method of multipliers (ADMM) [121]. Indeed, the DistPH method also follows a pattern identical to the distributed ADMM (D-ADMM) [122]–[124]. However, we shall note that while the D-ADMM-like algorithms are often developed for solving unconstrained consensus optimization problems and primarily based on the Lagrangian method, the DistPH method starts from two-stage stochastic programs which typically involve continuous or even mixed-integer constraints.*

5.3.2 Additional Considerations on Algorithm Initialization and Termination

Even though asymptotic convergence of the DistPH method with arbitrary initialization has been theoretically guaranteed by Proposition 5.3.1 and Corollary 5.3.1 (for the convex case). For real-world applications, however, appropriate choice of the algorithm initialization and termination criteria can greatly improve solution efficiency. Thus, in this subsection, we consider these two computational issues when implementing the DistPH method. Note that

the techniques introduced in this subsection can be applied to stochastic programs involving both continuous and/or mixed-integer decision variables.

As suggested in the literature [114], [115], including the one proposing the original PH method [78], an effective initialization scheme is to set the primal variables by solving the following subproblems,

$$(\mathbf{x}_i^0, \mathbf{y}_i^0) \in \underset{(\mathbf{x},\mathbf{y})\in\mathcal{K}_i}{\arg\min} \quad p_i(\mathbf{c}^\top \mathbf{x} + \mathbf{q}_i^\top \mathbf{y}), \tag{5.22}$$

and simply setting the dual variables as $\boldsymbol{\nu}_i^0 = 0$. It can be immediately verified that, since no coordination or variable sharing is needed, such an initialization scheme is readily implementable under a distributed framework.

For the termination criterion, unfortunately, a well-accepted approach which measures the following primal residual $\gamma^k \leq \varepsilon$ cannot be directly applied under a distributed framework,

$$\gamma^k := \sum_{i=1}^{I} p_i \|\mathbf{x}_i^k - \bar{\mathbf{x}}^k\|. \tag{5.23}$$

Note that $\bar{\mathbf{x}}^k := \sum_{i=1}^{I} p_i \mathbf{x}_i^k$ denotes the desired non-anticipativity solution and γ^k characterizes the overall distance between all $\mathbf{x}_i^k$'s and $\bar{\mathbf{x}}^k$ at each iteration k. Computing this primal residual in distributed settings mainly faces two challenges: i) $\bar{\mathbf{x}}^k$ is unknown for each agent i since it needs all the information over the entire network; and ii) calculating γ^k needs all the information as well, even though each agent can obtain its own $\bar{\mathbf{x}}^k$. Inherently, the two challenges can be attributed to the lack of global awareness by individual agents. To address the challenges, we build on the distributed sum-tracking technique [34]. The key idea is to assign each agent a new variable π_i^k, aiming at tracking the desired global information iteratively.

Suppose that a general time-varying in-stream data $\{\boldsymbol{\xi}_i^k\}_{k\in\mathbb{N}_+}$ is received by each agent i, we assign each agent a tracking variable $\boldsymbol{\pi}_i^k$ initialized by $\boldsymbol{\pi}_i^0 = I\boldsymbol{\xi}_i^0$, and let it be updated as

$$\boldsymbol{\pi}_i^{k+1} = \sum_{j\in\mathcal{N}_i} w_{ij}\boldsymbol{\pi}_j^k + I(\boldsymbol{\xi}_i^{k+1} - \boldsymbol{\xi}_i^k). \tag{5.24}$$

Since the update of $\boldsymbol{\pi}_i^{k+1}$ only requires information from node i itself and its neighbors, the iterative updating step (5.24) obeys the underlying graph $\mathcal{G}$, and thus can be implemented under a distributed framework. In addition, we require that the coefficients w_{ij}'s in (5.24) satisfy the double-stochasticity condition; see Assumption 1.4.12. As a consequence, the convergence of the distributed sum-tracking (5.24) can be established as follows.

Theorem 5.3.1. *Suppose that the time-invariant undirected graph $\mathcal{G}$ satisfies Assumption 1.4.8 and the prefixed weight matrix W satisfies Assumption 1.4.12. If the received in-stream data $\{\boldsymbol{\xi}_i^k\}_{k\in\mathbb{N}_+}$ for each agent is stable, i.e.,*

$$\lim_{k\to\infty} \|\boldsymbol{\xi}_i^{k+1} - \boldsymbol{\xi}_i^k\| = 0,\ \forall i \in \mathcal{I}, \tag{5.25}$$

then the sequence of tracking variables $\{\boldsymbol{\pi}_i^k\}_{k\in\mathbb{N}_+}$, generated by scheme (5.24), has the following convergence

$$\|\boldsymbol{\pi}_i^k - \sum_{i=1}^{I} \boldsymbol{\xi}_i^k\| \to 0,\ \forall i \in \mathcal{I}. \tag{5.26}$$

Proof. The proof follows a similar manner as the one for Theorem 1 in [36], and is thus omitted. □

With the help of the distributed sum-tracking technique as introduced above, we are now ready to compute the primal residual γ^k as follows. Let $\{p_i\mathbf{x}_i^k\}_{k\in\mathbb{N}_+}$ be the sequence of in-stream data received by each agent i, i.e., $\boldsymbol{\xi}_i^k = p_i\mathbf{x}_i^k$. Given that the stability requirement (5.25) of received data has already been guaranteed by the convergence of $\mathbf{x}_i^k$ (see Corollary 5.3.1), we know that the tracking variable $\boldsymbol{\pi}_i^k$ maintained by each agent i must have the convergence $\boldsymbol{\pi}_i^k \to \bar{\mathbf{x}}^k$. Therefore, each agent i will be able to gradually obtain $\bar{\mathbf{x}}^k$ by iteratively updating the tracking variable $\boldsymbol{\pi}_i^k$. Moreover, to compute the primal residual γ^k, we perform another distributed sum-tracking procedure by viewing the in-stream data as $\{p_i\|\mathbf{x}_i^k - \boldsymbol{\pi}_i^k\|\}_{k\in\mathbb{N}_+}$. Note that the stability of this set of data is guaranteed by the non-anticipativity constraints, i.e., $\mathbf{x}_i^k \to \bar{\mathbf{x}}^k$, $\forall i \in \mathcal{I}$. Thus, we introduce a new tracking variable $\hat{\pi}_i^k \in \mathbb{R}$ performing an outer-tracking procedure, then the convergence, described in (5.26),

ensures $\hat{\pi}_i^k \to \gamma^k$, $\forall i \in \mathcal{I}$. We summarize the distributed computing scheme for termination verification in Algorithm 7.

Algorithm 7: Distributed Computing for Termination Verification

Data: Given a tolerance ε, weight matrix W and $\mathbf{x}_i^0$. Let $\boldsymbol{\pi}_i^0 = \mathbf{x}_i^0$ and $\hat{\pi}_i^0 = 0$. Set initial iteration index $k = 0$.

while the criterion $|\hat{\pi}_i^k| < \varepsilon$ is NOT satisfied for $k > 0$ **do**

Each agent $i \in \mathcal{I}$ **simultaneously** does

(S.1) Updates its variable $\mathbf{x}_i^{k+1}$ by Algorithm 6.

(S.2) Receives $\boldsymbol{\pi}_j^k$ from the neighbors, then updates $\boldsymbol{\pi}_i^{k+1}$ by

$$\boldsymbol{\pi}_i^{k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} \boldsymbol{\pi}_j^k + Ip_i(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k),$$

and broadcasts $\boldsymbol{\pi}_i^{k+1}$.

(S.3) Receives $\hat{\pi}_j^k$ from the neighbors, then updates $\hat{\pi}_i^{k+1}$ by

$$\hat{\pi}_i^{k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} \hat{\pi}_j^k + Ip_i(\|\mathbf{x}_i^{k+1} - \boldsymbol{\pi}_i^{k+1}\| - \|\mathbf{x}_i^k - \boldsymbol{\pi}_i^k\|),$$

and broadcasts $\hat{\pi}_i^{k+1}$.

(S.4) Let $k \leftarrow k + 1$, and continue.

end

5.4 Mixed-Integer Case: Computation of the Lower Bound

In this section, we explore the extended applicability of the proposed DistPH method to two-stage stochastic programs with mixed-integer decision variables, i.e., SMIPs. In particular, we investigate the distributed technique for computing the Lagrangian-dual lower bounds.

While proposed originally in [78] for solving stochastic programs with only continuous decision variables, the basic PH method has already been practically adopted as a heuristic to deal with many SMIPs. Considering that there is no theoretical guarantee for convergence when applying the basic PH method to SMIPs, an effective way to evaluate the quality of the obtained solution is fairly important under such circumstances. Recently, a lower-bounding

technique is proposed in [115] for dealing with the SMIPs, which allows to assess the solution generated at each iteration of the basic PH method. Furthermore, it is shown that a sequence of lower bounds calculated by such technique converges to the best lower bound which is as tight as the one obtained by dual decomposition methods [79], [125], [126]. Here, we aim at developing a distributed technique for computing the low bounds under the peer-to-peer computing network.

Before presenting our distributed lower-bounding technique, i.e., distributed Lagrangian dual computation, we first introduce the centralized lower-bounding technique proposed in [115]. Considering an SMIP formulation (5.4), its Lagrangian dual function by applying Lagrangian relaxation to the non-anticipativity constraints reads

$$\mathcal{D}(\boldsymbol{\omega}) = \min_{\substack{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{K}_i \\ \mathbf{z}\in\mathbb{R}^{p_1}}} \Big\{ \sum_{i=1}^{I} p_i \Big(\mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i + \boldsymbol{\omega}_i^\top (\mathbf{x}_i - \mathbf{z}) \Big) \Big\}, \tag{5.27}$$

where the dual variable $\boldsymbol{\omega} \in \mathbb{R}^{Ip_1}$ stacks all individual $\boldsymbol{\omega}_i$'s. To decompose the minimization problem (5.27) into independent components for each scenario, the restriction on dual variables $\boldsymbol{\omega}_i$ with $\sum_{i=1}^{I} p_i \boldsymbol{\omega}_i = 0$ is incorporated. Thus, variable $\mathbf{z}$ in (5.27) is eliminated and the Lagrangian dual function can be rewritten as

$$\mathcal{D}(\boldsymbol{\omega}) = \sum_{i=1}^{I} p_i \mathcal{D}_i(\boldsymbol{\omega}_i), \tag{5.28}$$

where each $\mathcal{D}_i(\boldsymbol{\omega}_i)$ has

$$\mathcal{D}_i(\boldsymbol{\omega}_i) = \min_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{K}_i} \Big\{ \mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i + \boldsymbol{\omega}_i^\top \mathbf{x}_i \Big\}. \tag{5.29}$$

It is well-known that the best lower bound for problem (5.4) can be achieved by solving the following maximization problem

$$\zeta^\star = \max_{\boldsymbol{\omega}} \ \mathcal{D}(\boldsymbol{\omega}) \quad \text{subject to} \quad \sum_{i=1}^{I} p_i \boldsymbol{\omega}_i = 0. \tag{5.30}$$

Many approaches are developed in the literature for solving (5.30). One of them, developed under the framework of the PH method [115], considers the following convexified form of (5.4),

$$\begin{aligned} \min_{\mathbf{x}_i,\mathbf{y}_i,\mathbf{z}} \quad & \sum_{i=1}^{I} p_i(\mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i), \\ \text{s. t.} \quad & \mathbf{x}_i = \mathbf{z}, \\ & (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}_i, \ \forall i \in \mathcal{I}, \end{aligned} \tag{5.31}$$

where $\mathcal{C}_i = \mathbf{Conv}(\mathcal{K}_i)$ denotes the convex hull of the feasible set $\mathcal{K}_i$. As a result, the best lower bound $\zeta^\star$ can be obtained by iteratively performing the following computation,

$$\zeta^k = \sum_{i=1}^{I} p_i \mathcal{D}_i^c(\boldsymbol{\omega}_i^k), \tag{5.32}$$

where $\mathcal{D}_i^c$ is defined as

$$\mathcal{D}_i^c(\boldsymbol{\omega}_i^k) := \min_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{C}_i} \left\{ \mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i + (\boldsymbol{\omega}_i^k)^\top \mathbf{x}_i \right\}, \tag{5.33}$$

and $\boldsymbol{\omega}_i^k$ is the dual variable obtained by the dual update step of the PH method.

Recall that the problem of interest here is to perform the computation of (5.32) in a distributed manner. There are two key issues need to be resolved: i) computing ζ^k by (5.32) requires information from all agents; and ii) the dual variable $\boldsymbol{\omega}_i^k$ is not accessible during the procedure of our DistPH method. Here, we first investigate the second issue, and show that the accessible $\boldsymbol{\nu}_i^k$ is also able to provide a viable way to compute lower bounds. The computation of ζ^k from a centralized perspective, involving $\boldsymbol{\nu}_i^k$, can be performed as

$$\zeta^k = \sum_{s=1}^{m} p_i \mathcal{D}_i^+(\boldsymbol{\nu}_i^k), \tag{5.34}$$

where

$$\mathcal{D}_i^+(\boldsymbol{\nu}_i^k) := \min_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{C}_i} \left\{ \mathbf{c}^\top \mathbf{x}_i + \mathbf{q}_i^\top \mathbf{y}_i + (d_i/p_i)(\boldsymbol{\nu}_i^k)^\top \mathbf{x}_i \right\}. \tag{5.35}$$

We validate the computation (5.34) and (5.35) by the following theorem.

Theorem 5.4.1. *Suppose that the sequence* $\{\zeta^k\}_{k\in\mathbb{N}_+}$ *is generated by the computation in* (5.34) *and* (5.35)*, with* $\{\boldsymbol{\nu}_i^k\}_{k\in\mathbb{N}_+}$ *generated by applying Algorithm 6 to problem* (5.31)*. As a result, each* ζ^k *is a lower bound of the optimal objective value of the problem* (5.4)*, and the convergence of sequence* $\{\zeta^k\}_{k\in\mathbb{N}_+}$ *has*

$$\zeta^k \to \zeta^\star. \tag{5.36}$$

Proof. We first show that, if $\boldsymbol{\nu}_i$ is initialized as $\boldsymbol{\nu}_i^0 = 0,\ \forall i \in \mathcal{I}$, then the updating scheme (5.20) of $\boldsymbol{\nu}_i^k$ implies the following result,

$$\sum_{i=1}^{I} d_i \boldsymbol{\nu}_i^k = 0, \quad \forall k \in \mathbb{N}_+. \tag{5.37}$$

In fact, by (5.19) and (5.20), it can be seen that

$$\begin{aligned}\sum_{i=1}^{I} d_i \boldsymbol{\nu}_i^{k+1} &= \sum_{i=1}^{I} d_i \boldsymbol{\nu}_i^k + \rho \sum_{i=1}^{I} (\mathbf{x}_i^{k+1} - \hat{\mathbf{z}}_i^{k+1}) \\ &= \sum_{i=1}^{I} d_i \boldsymbol{\nu}_i^k + \frac{\rho}{2}\Big(\sum_{i=1}^{I} \mathbf{x}_i^{k+1} - \sum_{i=1}^{I} \frac{1}{d_i} \sum_{j\in\mathcal{N}_i} \mathbf{x}_j^{k+1} \Big)\end{aligned} \tag{5.38}$$

It can be immediately verified that the second term in the last equation must be zero, and thus (5.37) is true when the initialization is zero.

Now, we prove that each ζ^k is a valid lower bound, i.e., $\zeta^k \le \chi^\star,\ \forall k \in \mathbb{N}_+$ where $\chi^\star$ is the optimal function value for the SMIP formulation (5.4). Let us denote $(\mathbf{x}_i^\star, \mathbf{y}_i^\star)$ and $\mathbf{z}^\star$ as the optimal solution to (5.4), then we have

$$\chi^\star = \sum_{i=1}^{I} p_i (\mathbf{c}^\top \mathbf{x}_i^\star + \mathbf{q}_i^\top \mathbf{y}_i^\star), \tag{5.39}$$

and

$$\mathbf{z}^\star = \mathbf{x}_i^\star, \quad \forall i \in \mathcal{I}. \tag{5.40}$$

Given that the feasible set restricted by $\mathcal{K}_i$ is always a subset of its convex hull $\mathcal{C}_i$, i.e., $(\mathbf{x}_i^\star, \mathbf{y}_i^\star) \in \mathcal{C}_i$, we have,

$$\begin{aligned}
\zeta^k &= \sum_{s=1}^{m} p_i \mathcal{D}_i^+(\boldsymbol{\nu}_i^k) \\
&\overset{(5.1.a)}{\leq} \sum_{i=1}^{I} p_i \Big(\mathbf{c}^\top \mathbf{x}_i^\star + \mathbf{q}_i^\top \mathbf{y}_i^\star + (d_i/p_i)(\boldsymbol{\nu}_i^k)^\top \mathbf{x}_i^\star \Big) \\
&\overset{(5.1.b)}{=} \sum_{i=1}^{I} p_i (\mathbf{c}^\top \mathbf{x}_i^\star + \mathbf{q}_i^\top \mathbf{y}_i^\star) + (\sum_{i=1}^{I} d_i \boldsymbol{\nu}_i^k)^\top \mathbf{z}^\star \\
&\overset{(5.1.c)}{=} \sum_{i=1}^{I} p_i (\mathbf{c}^\top \mathbf{x}_i^\star + \mathbf{q}_i^\top \mathbf{y}_i^\star) \\
&= \chi^\star.
\end{aligned} \tag{5.41}$$

Note that inequality (5.1.a) is by the definition (5.35) of function $\mathcal{D}_i^+(\boldsymbol{\nu}_i^k)$; equality (5.1.$b$) follows from (5.40); and equality (5.1.c) follows from (5.37).

To show the convergence of ζ^k, we leverage the proved convergence result of Algorithm 6. Given that the DistPH method is applied to problem (5.31) which is guaranteed to be convex, based on the proof of Theorem 5.1 in [78], we know that variable $\boldsymbol{\nu}_i^k$ converges to an optimal dual $\mathbf{v}^\star$. In addition, it can be verified that the primal optimal $\mathbf{w}^\star$ and the dual optimal $\mathbf{v}^\star$ compose a saddle point of the following function

$$\kappa(\mathbf{w}, \mathbf{v}) := F(\mathbf{w}) + \mathbf{v}^\top \mathbf{w}, \tag{5.42}$$

subject to the minimization with respect to $\mathbf{w}$ and the maximization with respect to $\mathbf{v}$. By definition of the overall objective function $F(\mathbf{w})$ (see equation (5.10)), the convergence of $\zeta^k \to \zeta^\star$ straightforwardly follows the convergence of dual variable $\boldsymbol{\nu}_i^k$ to $\mathbf{v}^\star$. □

By Theorem 5.4.1, we have already found a viable way to compute the lower bounds from a centralized perspective. That is, each agent solves a minimization problem (5.35) by using the updated dual variable $\boldsymbol{\nu}_i^k$, then the computation of weighted average, as shown in (5.34), gives a lower bound ζ^k at the current iteration k. Now, the problem becomes how to compute the weighted average under the distributed framework. Here, we recall the

distributed sum-tracking technique as introduced in the previous Section 5.3.2, and perform the following tracking step,

$$\pi_i^{k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} \pi_j^k + Ip_i \Big(\mathcal{D}_i^+(\boldsymbol{\nu}_i^{k+1}) - \mathcal{D}_i^+(\boldsymbol{\nu}_i^k) \Big). \tag{5.43}$$

Note that the stability of sequence $\{\mathcal{D}_i^+(\boldsymbol{\nu}_i^k)\}_{k \in \mathbb{N}_+}$ can be guaranteed by the convergence of Algorithm 6. According to both Theorems 5.4.1 and Corollary 5.3.1, we conclude that sequence $\{\pi_i^k\}_{k \in \mathbb{N}_+}$ has the following convergence

$$\pi_i^k \to \zeta^\star, \quad \forall i \in \mathcal{I}. \tag{5.44}$$

5.5 Numerical Results

5.5.1 Simulation Study Setup

We test our DistPH method with two benchmark stochastic programming instances, i.e., the `SIZES` and `SSLP` problems, both of which are publicly available in SIPLIB [127]. We implement the DistPH method by using Python programming language under the framework of PySP [116], which is an open-source software package and enables the stochastic programming extension of Pyomo (Python Optimization Modeling Objects) [117] for modeling and solving stochastic programs. Specifically, PySP includes an implementation of the basic (serial) PH method and its parallel variant by using a Pyro (Python Remote Objects) package to manage communications between different threads or nodes. We use PySP for stochastic programming model implementation and adopt Pyro for the communications management. It should be noted that we make our own implementation of the DistPH method and its extension for SMIP. A primary difference between our implementation and existing parallel implementations is that we do not assume a master node that computes the weighted-average and passes the information onto all workers. Instead, we only rely on peer communications under a peer-to-peer multi-agent network. The subproblems at each iteration are solved with Gurobi 9.1 via the inherent quadratic solver engine. All computations in our work are performed on a computing server with two Intel Xeon E5-2690 (32 cores)

running at 2.90GHz with 256GB of RAM. Notice that while the parallel and distributed PH methods are implemented on multiple cores according to the number of scenarios, the basic PH procedure is performed on a single core of the server.

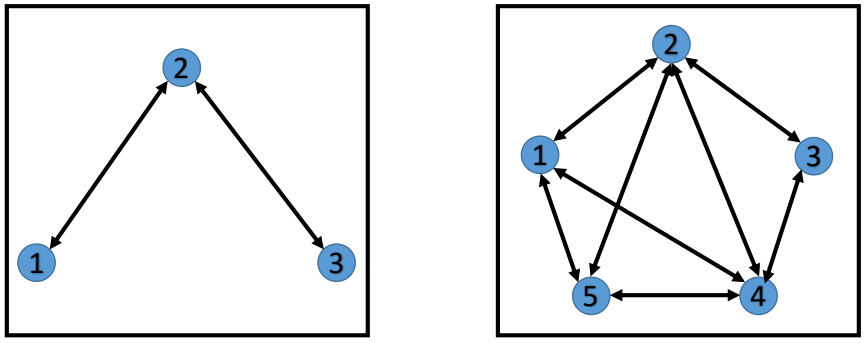
5.5.2 The SIZES Instances

The SIZES problem refers to a multi-stage (multi-period) stochastic lot-sizing problem [128]. In this simulation, we consider three instances of the SIZES problem, i.e., SIZES3 (with 3 scenarios), SIZES5 (with 5 scenarios), and SIZES10 (with 10 scenarios). Note that mixed-integer decision variables appear in both stages of the instances. The characteristics of their extensive forms are summarized in Table 5.1.

Table 5.1. Extensive form characteristics of the SIZES instances

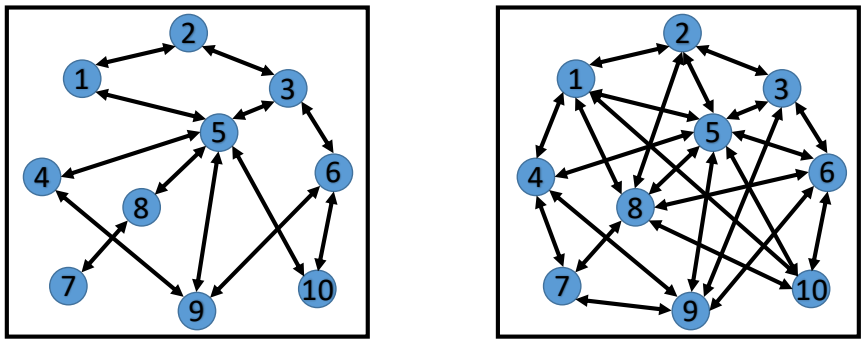
instance	# of scenarios	# of binaries	# of variables	# of constraints
SIZES3	3	40	300	124
SIZES5	5	60	450	186
SIZES10	10	110	825	341

We first evaluate the solution performance of three variants of the PH method, namely the basic PH method (B-PH), the parallel PH method (P-PH) and our DistPH method (D-PH). For the P-PH method, the underlying network follows the specific master-worker architecture. For the D-PH method, we consider different degrees of connectivity of the underlying peer-to-peer network, and use the algebraic connectivity to quantify the degree of connectivity. The algebraic connectivity, denoted by σ, is calculated by the second smallest eigenvalue of the graph Laplacian matrix L, and its magnitude is known to reflect how well-connected the underlying graph is, i.e., a sparser network has a smaller σ. This metric has been widely used in the study of distributed algorithms and analysis of network properties; see e.g. [129], [130]. In addition to a fully-connected network (each node has access to any other ones), we assess the D-PH method on networks with lower algebraic connectivity, as shown in Fig. 5.1. To objectively compare the solution performance of the three methods,



(a) network for `SIZES3`
$\sigma = 1.0$

(b) network for `SIZES5`
$\sigma = 2.0$



(c) network for `SIZES10`
$\sigma = 0.415$

(d) network for `SIZES10`
$\sigma = 2.255$

Figure 5.1. Communication topology of multi-agent computing networks

we disable the termination threshold by setting $\varepsilon = 0$ in Algorithm 7 and run each method for the same number of 200 iterations.

The results for all the methods after 200 iterations are presented in Table 5.2. Note that we term the D-PH method with a fully-connected network as D-PH (F-Conn) in the table. The reference objective values for all three instances are provided in SIPLIB. It is observed from Table 5.2 that, while the objective value obtained via our D-PH method is comparable to the B-PH method, yet the execution time is significantly reduced since multiple computing nodes are utilized. More precisely, the scalability of the D-PH method is shown in Fig. 5.2. It is observed that considerable speed-up can be achieved by the distributed framework. Moreover, compared to the P-PH method provided in PySP, the D-PH method also obtains a similar performance in terms of both execution time and final objective value. In fact, we have to remark here that the P-PH method and our D-PH method are not quite comparable for the following reasons. First, owing to central coordination provided by the master node,

Table 5.2. Solution performance of the SIZES instances (with 200 iterations)

Instance	Method	Exec Time	Objective	Ref Objective
SIZE3	B-PH	155.74s	225180.0280 (0.33%)	224434.32
	P-PH	66.99s	225180.0280 (0.33%)	
	D-PH (F-Conn)	69.42s	225139.3112 (0.31%)	
	D-PH ($\sigma = 1.0$)	68.28s	225263.9379 (0.37%)	
SIZES5	B-PH	259.87s	226483.5898 (0.89%)	224486.00
	P-PH	74.21s	226483.5898 (0.89%)	
	D-PH (F-Conn)	80.85s	226687.2803 (0.98%)	
	D-PH ($\sigma = 2.0$)	75.12s	226535.7782 (0.91%)	
SIZES10	B-PH	582.35s	226513.0279 (0.87%)	224564.30
	P-PH	98.57s	226513.0279 (0.87%)	
	D-PH (F-Conn)	118.21s	226660.3938 (0.93%)	
	D-PH ($\sigma = 2.255$)	109.17s	226698.7066 (0.95%)	
	D-PH ($\sigma = 0.415$)	102.80s	226452.6490 (0.84%)	

the master-worker architecture indeed offers an ideal architecture to realize parallelization. If the communication cost is measured by the number of communication channels, the P-PH method also incurs lower communication cost compared to the D-PH method. However, one should note that, under a master-worker architecture, all the communication burden goes to the master node, and thus it easily becomes a communicating bottleneck for the entire network. In contrast, since the distributed method builds on a peer-to-peer network and each agent only needs to communicate with its neighbors, the communication burden is thus spread over the network and each individual agent only contributes to a portion of the communication cost. In addition, for many real-world in-network computing tasks, due to insufficient communication bandwidth and storage capacity, it is highly likely that no single node can take on such a heavy coordination task. In conclusion, we argue that our D-PH method is more flexible and applicable in real-world applications, as it can be applied with the general connected computing network.

Apart from the comparison between the different variants of the PH methods, we further evaluate the performance of the D-PH method with different degrees of algebraic connectivity of the network. In this simulation study, we enable the termination criterion by setting the

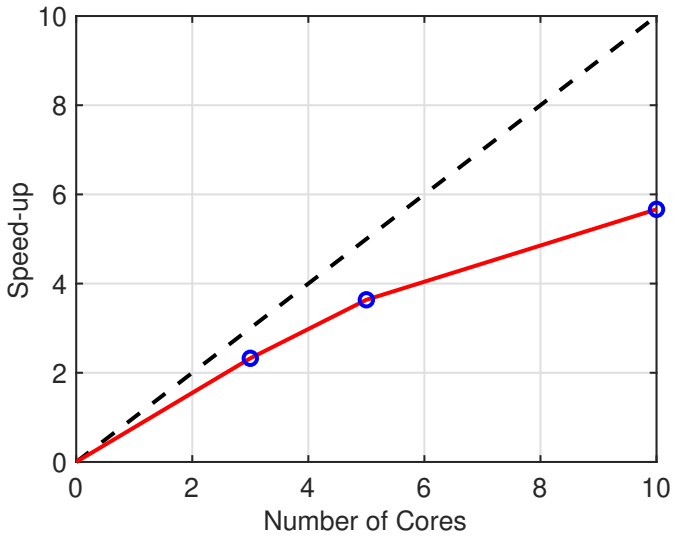


Figure 5.2. Scalability of the D-PH method in solving SIZES problems

Table 5.3. Solution performance of the SIZES instances (by termination criterion)

Instance	Method	# of Iters	Objective	Ref Objective
SIZE3	P-PH	77	225474.9947 (0.46%)	224434.32
	D-PH (F-Conn)	20	226085.2030 (0.74%)	
	D-PH ($\sigma = 1.0$)	140	225622.5662 (0.53%)	
SIZE5	P-PH	200	226483.5898 (0.89%)	224486.00
	D-PH (F-Conn)	65	227110.3133 (1.17%)	
	D-PH ($\sigma = 2.0$)	200	226535.7782 (0.91%)	
SIZE10	P-PH	200	226513.0279 (0.87%)	224564.30
	D-PH (F-Conn)	72	226770.7423 (0.98%)	
	D-PH ($\sigma = 2.255$)	200	226698.7066 (0.95%)	
	D-PH ($\sigma = 0.415$)	200	226452.6490 (0.84%)	

threshold $\epsilon = 10^{-4}$, and the algorithm will terminate once the primal residual is less than the threshold. It is suggested by Tables 5.3 that a larger number of iterations are required to reach the same termination threshold under a network with smaller algebraic connec-

tivity. Interestingly, it is also observed from the table that the D-PH method under the fully-connected network reaches the termination condition with a fewer number of iterations compared to the P-PH method. This is due to the differences between the two methods as noted in Remark 1.

5.5.3 The SSLP Instances

For the SSLP, a two-stage stochastic server location problem [131], we consider two instances, i.e., sslp_5_25_50 with 50 scenarios and sslp_5_25_100 with 100 scenarios. The characteristics of these two instances are listed in Table 5.4. We again investigate the solution performance of three variants of the PH method. In this simulation study, since these SSLP instances involve a larger number of scenarios than the SIZES instances studied, we create a network consisting of 25 computing nodes, and thus two or four scenarios are bundled together at the each node. Same as before, the computing networks are specified with different degrees of algebraic connectivity σ.

Table 5.4. Extensive form characteristics of the SSLP instances

instance	# of scenarios	# of binaries	# of variables	# of constraints
sslp_5_25_50	50	6255	250	1501
sslp_5_25_100	100	12505	500	3001

The numerical results for all three variants of the PH methods are presented in Table 5.5, when we set the threshold as $\epsilon = 0$ and terminate the execution after the 200-th iteration. The similar observations can be obtained as for the SIZES instances. However, as distinct from the previous example, it can be seen from Table 5.5 that almost all methods (except D-PH over the sparsest network) can obtain the solution with 100% accuracy. More importantly, the D-PH method (and also P-PH method) achieves a nearly linear speed-up of the execution time. In addition, to further examine the influence of network topology on our D-PH method, we set the termination threshold as $\epsilon = 10^{-4}$, and the numerical results obtained are shown in Table 5.6. It can be verified that a larger number of iterations are demanded by the D-PH

method when the underlying network has a sparser topology. Nevertheless, in some cases, e.g., the instance `sslp_5_25_50`, the D-PH method over a non-fully-connected network still exhibits faster convergence than the P-PH method.

Table 5.5. Solution performance of the `SSLP` instances (with 200 iterations)

Instance	Method	Exec Time	Objective	Ref Objective
`sslp_5_25_50`	B-PH	1567.2s	-121.60	-121.60
	P-PH	84.0s	-121.60	
	D-PH (F-Conn)	93.4s	-121.60	
	D-PH ($\sigma = 22.786$)	89.8s	-121.60	
	D-PH ($\sigma = 1.686$)	87.2s	-121.60	
`sslp_5_25_100`	B-PH	3683.4s	-127.37	-127.37
	P-PH	166.2s	-127.37	
	D-PH (F-Conn)	175.2s	-127.37	
	D-PH ($\sigma = 24.882$)	171.6s	-127.37	
	D-PH ($\sigma = 3.162$)	168.3s	-125.59 (1.4%)	

Table 5.6. Solution performance of the `SSLP` instances (by termination criterion)

Instance	Method	# of Iters	Exec Time	Objective
`sslp_5_25_50`	P-PH	68	30.82s	-121.60
	D-PH (F-Conn)	31	14.31s	-121.60
	D-PH ($\sigma = 22.786$)	67	29.96s	-121.60
	D-PH ($\sigma = 1.686$)	135	59.94s	-121.60
`sslp_5_25_100`	P-PH	77	63.75s	-127.37
	D-PH (F-Conn)	49	44.54s	-127.37
	D-PH ($\sigma = 24.882$)	119	105.76s	-127.37
	D-PH ($\sigma = 3.162$)	200	176.19s	-125.59

6. APPLICATIONS WITH MULTI-UAV SYSTEMS

Recently, unmanned aerial vehicles (UAVs) have been widely adopted in various applications, including environment/health monitoring [132], [133], package delivery [134], [135], surveillance [136]–[138], etc. The popularity of UAVs can be attributed to their remarkable features, such as mobility, flexibility, and accessibility. In particular, it is often beneficial when a team of coordinated UAVs is employed for some mission, rather than one single UAV. For instance, by utilizing the cooperation among multiple UAVs, such a coordinated team has the potential to accomplish missions whose total computational workload goes beyond the capability of each single UAV. On the other hand, the task allocation and cooperation protocol are supposed to be specifically devised under such circumstances. This also leads to the popularity of paradigm of distributed processing over the last decades. Motivated by this, in this chapter we investigate two real-world applications involved with the multi-UAV systems: i) distributed data fusion for on-scene signal sensing; and ii) distributed source tracking in unknown environments.

6.1 Application I: Distributed Data Fusion for On-Scene Signal Sensing

In the first application, we consider a healthcare scenario where the health status of certain target in rural environment needs to be monitored through sensing some on-scene biological signals, during or after an incident [139]. In particular, we focus on a distributed data fusion problem, i.e., integrating information from various measurements of individual UAVs. It is assumed that each UAV is capable of sensing signals, performing computational tasks, and exchanging information with its immediate neighbors. Based on the limited local measurement and exchanged information only from neighbors, we aim to develop a distributed algorithmic framework to fuse the information gathered by the entire UAV network. It is expected that such a multi-UAV system benefits from the accuracy of data gained from multiple sensors, while keeping computational load favorable for each single UAV since it only needs to process a relatively small set of data. The ultimate goal is to enforce the UAVs to reach consensus at which each UAV individually provides a more reliable fused sensing result with higher accuracy, compared to the local measurement collected by the UAV itself.

6.1.1 Sensor Fusion Model

Suppose that the target signal is represented by a time-series $s_0(t)$, we formulate the continuous-time ill-polluted measurement $m_i(t)$ by each individual UAV $i \in \mathcal{I} = \{1, 2, \cdots, I\}$ as follows,

$$m_i(t) := \gamma_i(t)s_0(t) + b_i(t) + n_i(t). \tag{6.1}$$

Here, $\gamma_i(t)$ and $b_i(t)$ are the scaling factor and offset bias, respectively; and $n_i(t)$ denotes the white Gaussian measurement noise. We remark that such a measurement model (6.1) has been widely adopted in the existing literature, such as the studies of multi-sensor multi-target tracking[140], [141], sensor calibration for mobile sensing [142], [143], and data fusion for body sensors [144]. In particular, the influence of scaling factor $\gamma_i(t)$ and offset bias $b_i(t)$ in sensor measurement is intensively investigated in [143]. While the offset bias could be caused by sensor misalignment, the scaling factor usually occurs in some UAV-related sensor measurement such as accelerometer and gyroscope [145]. It is also emphasized that such an ill-polluted measurement model can be hardly handled by the standard least-square estimation or maximum-likelihood estimation, due to the existence of the unknown offset bias $b_i(t)$. To deal with this issue, we apply the criterion of maximizing signal-to-noise ratio (MSNR), inspired by[144]. In addition, the method developed here is more applicable, as it does not require the statistics of white Gaussian noise $n_i(t)$ to be known.

An alternative discrete-time model of measurement (6.1) can be obtained by discretizing the data into a d-dimensional vector $\mathbf{m}_i := [m_i(1), m_i(2), \cdots, m_i(t), \cdots, m_i(d)]^\top \in \mathbb{R}^d$ with

$$\mathbf{m}_i := \Gamma_i \mathbf{s}_0 + \mathbf{b}_i + \mathbf{n}_i, \tag{6.2}$$

where $\Gamma_i \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_i \in \mathbb{R}^d$ are the discretized scaling factor and offset bias, respectively; $\mathbf{n}_i \in \mathbb{R}^d$ is the measurement noise; and $\mathbf{s}_0 \in \mathbb{R}^d$ represents the discretized true signal. Note that, while we here use t to denote the index for both continuous time-step and discrete time-step, we will only focus on the discrete-time case in the following analysis.

Keep in mind that our objective is to distributively recover the true signal $\mathbf{s}_0$ from a set of I ill-polluted measurements $M := [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_I]^\top \in \mathbb{R}^{I\times d}$. Since the measurement noise $\mathbf{n}_i$ is assumed to have the nature of white Gaussian and it will be later eliminated by an MSNR procedure, let us first focus on the scaling factor Γ_i and offset bias $\mathbf{b}_i$. In order to cope with the effects caused by Γ_i and $\mathbf{b}_i$, a natural idea is to globally seek a group of re-scaling coefficients $\boldsymbol{\alpha} := [\alpha_1, \alpha_2, \cdots, \alpha_I]^\top \in \mathbb{R}^I$ and anti-offset coefficients $\boldsymbol{\beta} := [\beta_1, \beta_2, \cdots, \beta_I]^\top \in \mathbb{R}^I$, such that the filtered signals

$$\mathbf{s}_i := \alpha_i(\mathbf{m}_i - \beta_i \mathbf{1}_d), \quad i \in \mathcal{I}, \tag{6.3}$$

will provide better estimates of the true signal $\mathbf{s}_0$. The underlying intuition for estimating $\mathbf{s}_i$ is using α_i and β_i to compensate the measurement error caused by Γ_i and $\mathbf{b}_i$. We remark that developing a scheme to seek $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in a distributed manner faces two main challenges: i) rather than using the limited local $\mathbf{m}_i$ to independently estimate α_i and β_i, we are expected to use global information M to cooperatively obtain the better estimates; and ii) the lack of global information for each individual UAV is a key issue on the other hand, since it can only access the local $\mathbf{m}_i$ and a limited number of neighbors.

To tackle these two difficulties, we develop a framework with two stages correspondingly: i) we stand in a global view to introduce a centralized algorithm for solving the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ by leveraging the idea of MSNR, see Section 6.1.2; and 2) by leveraging the average consensus scheme developed in Section 2.4 in Chapter 2, we design a distributed protocol to locally force each UAV to reach the consensus at the optimality of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, see Section 6.1.3.

6.1.2 Centralized Optimal Averaging

Recall that our first issue is to obtain the optimal $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, such that the filtered signals $\mathbf{s}_i$ defined in (6.3) will be able to provide the best estimates. To this end, we introduce an optimal averaging method [146] by maximizing signal-to-noise ratio (SNR),

$$\max_{\boldsymbol{\alpha},\,\boldsymbol{\beta}} \quad \text{SNR}, \tag{6.4}$$

and define the value of SNR as follows,

$$\text{SNR} := \frac{E_s}{E_n}. \tag{6.5}$$

In (6.5), E_s denotes the total energy of average signal estimate, i.e., $\bar{\mathbf{s}} := (1/I) \cdot \sum_{i=1}^{I} \mathbf{s}_i$, and E_s is computed as,

$$E_s := \|\bar{\mathbf{s}} - (1/d) \cdot \mathbf{1}_d \mathbf{1}_d^\top \bar{\mathbf{s}}\|^2. \tag{6.6}$$

E_n represents the total energy of residue noise,

$$E_n := \sum_{i=1}^{I} \left\|\mathbf{s}_i - \bar{\mathbf{s}}\right\|^2. \tag{6.7}$$

It can be proved that the optimal solutions $\boldsymbol{\alpha}^\star$ and $\boldsymbol{\beta}^\star$ to the maximization problem (6.4) can be derived independently and analytically, given in the following two propositions.

Proposition 6.1.1. *For any set of the re-scaling coefficients $\boldsymbol{\alpha}$, the optimal anti-offset coefficients $\boldsymbol{\beta}^\star := [\beta_1^\star, \beta_2^\star, \cdots, \beta_I^\star]^\top$ that maximize the SNR defined in (6.5), are given by,*

$$\beta_i^\star = \bar{m}_i := \frac{1}{d} \cdot \sum_{t=1}^{d} m_i(t), \quad i \in \mathcal{I}. \tag{6.8}$$

Proof. The proof can be found in Theorem 1 in [146]. □

Proposition 6.1.2. *For any set of the anti-offset coefficients $\boldsymbol{\beta}$, the optimal re-scaling coefficients $\boldsymbol{\alpha}^\star := [\alpha_1^\star, \alpha_2^\star, \cdots, \alpha_I^\star]^\top$ that maximize the SNR defined in (6.5), are given by the first generalized eigenvector of the characteristic equation,*

$$R\boldsymbol{\alpha}^\star = \lambda D \boldsymbol{\alpha}^\star, \tag{6.9}$$

where $R = [r_{ij}]_{i,j=1}^{I} \in \mathbb{R}^{I \times I}$ is the centered inner product matrix with $r_{ij} := \mathbf{m}_i^\top \mathbf{m}_j - d \cdot \bar{m}_i \bar{m}_j$, and D is the diagonal matrix of R, i.e., $D := \text{diag}\{r_{11}, r_{22}, \cdots, r_{II}\} \in \mathbb{R}^{I \times I}$.

Proof. The proof can be found in Theorem 2 in [146]. □

By Propositions 6.1.1 and 6.1.2, we have obtained an effective way to compute the optimal $\boldsymbol{\alpha}^\star$ and $\boldsymbol{\beta}^\star$ from a global perspective. Before continuing, let us make a few remarks here. First, as mentioned before, $\boldsymbol{\alpha}$ acts as a group of re-scaling weights as shown in (6.3), and thus scaling $\boldsymbol{\alpha}$ with any constant $\delta > 0$ will not change the value of SNR. This observation can be also verified by Proposition 6.1.2. In fact, $\boldsymbol{\alpha}^\star$ is computed as the generalized eigenvector of matrix pair (R, D), and a scaled $\delta\boldsymbol{\alpha}^\star$ with any $\delta > 0$ is still a valid eigenvector. Here, we require δ to be positive since $\delta\boldsymbol{\alpha}^\star$ serves as a group of weights. Second, given that each $\alpha_i^\star$ can be viewed as a weight for the local measurement $\mathbf{m}_i$, it is expected to obtain a relatively small $\alpha_i^\star$ when the measurement is far from the true signal $\mathbf{s}_0$. Based on these two reasons above, with restricting $\mathbf{1}_I^\top \boldsymbol{\alpha}^\star = 1$, we can finally obtain a global optimal recovery $\mathbf{s}^\star$ of the true signal as a summation of each $\mathbf{s}_i$, i.e.,

$$\mathbf{s}^\star := \sum_{i=1}^{I} \mathbf{s}_i = \sum_{i=1}^{I} \alpha_i^\star (\mathbf{m}_i - \beta_i^\star \mathbf{1}_d). \tag{6.10}$$

6.1.3 Distributed Sensor Fusion

We now consider a distributed algorithm to implement the above optimal averaging method. Recall that the desired distributed algorithm requires each individual UAV to reach a consensus point which is expected to be the same as $\mathbf{s}^\star$, given only the local measurement and exchanged information from neighbors. It is clear that the optimal anti-offset coefficients $\boldsymbol{\beta}^\star$ can be automatically computed in a distributed way, since each $\beta_i^\star$ only relies on the local measurement $\mathbf{m}_i$, as shown in (6.8). Moreover, suppose that $\alpha_i^\star$ and $\beta_i^\star$ have already been locally obtained by each individual UAV i, the final step which takes the summation as shown in (6.10) can also be simply implemented via a standard sum (or average) consensus procedure. Next, we present the distributed sum consensus scheme by leveraging the average consensus introduced in Chapter 2, and then develop the distributed algorithm to compute the optimal re-scaling coefficient $\boldsymbol{\alpha}^\star$ via the sum consensus scheme. Note that here we focus on the sum consensus scheme, instead of average consensus. By doing so, each UAV does not need to know the number of UAV I as a prior knowledge. This also make our algorithm more applicable in real-world applications.

Let us recall that the average consensus scheme developed in Chapter 2 enables each agent to asymptotically reach the average of their initial states, i.e.,

$$\lim_{k\to\infty} \|\mathbf{x}_i^k - \frac{1}{I}\cdot\sum_{i=1}^{I}\mathbf{x}_i^0\| = 0, \quad \forall i \in \mathcal{I}, \tag{6.11}$$

where we denote $\mathbf{x}_i^0$ the initial state of each agent i. Here, consider that the recovered signal $\mathbf{s}^\star$ is calculated by the summation of $\mathbf{s}_i$'s as shown in (6.10), rather than the average, therefore we are alternatively interested in a sum consensus scheme which requires the consensus point to be the summation of the initial states, i.e.,

$$\lim_{k\to\infty} \|\mathbf{x}_i^k - \sum_{i=1}^{I}\mathbf{x}_i^0\| = 0, \quad \forall i \in \mathcal{I}. \tag{6.12}$$

Following the same path as introduced in Section 2.4 Chapter 2, we present the sum consensus scheme as below. At the k-th iteration, each UAV i performs the following update of its local states $\mathbf{x}_i^k \in \mathbb{R}^d$ and $\boldsymbol{\phi}_i^k \in \mathbb{R}^I$,

$$\begin{aligned}
\boldsymbol{\phi}_i^{k+1} &= \sum_{j\in\mathcal{N}_{i,\text{in}}} w_{ij}\boldsymbol{\phi}_j^k, \\
\mathbf{x}_i^{k+1} &= \sum_{j\in\mathcal{N}_{i,\text{in}}} w_{ij}\mathbf{x}_j^k + \left(\frac{1}{\mathcal{I}_i\left(\boldsymbol{\phi}_i^{k+1}\right)} - \frac{1}{\mathcal{I}_i\left(\boldsymbol{\phi}_i^k\right)}\right)\mathbf{x}_i^0,
\end{aligned} \tag{6.13}$$

where the operator $\mathcal{I}_i(\cdot) : \mathbb{R}^I \to \mathbb{R}$ selects the i-th component of the input vector and the initialization of $\boldsymbol{\phi}_i^k$ is specified as $\boldsymbol{\phi}_i^0 = \mathbf{e}_i$. Note that we here consider the multi-UAV network to follow a time-invariant directed graph, and the set $\mathcal{N}_{i,\text{in}}$ denotes the i-th agent's in-neighborhood. In addition, the compliant weight matrix $W = [w_{ij}]_{i,j=1}^I$ is assumed to be row-stochastic; see Assumptions 1.4.9 and 1.4.10 in Chapter 1, and thus the benefits as stated at the beginning of Section 2.4 can be also achieved by the multi-UAV network considered in this application.

As an important building block for the complete distributed data fusion framework, we abstract the above sum consensus scheme as an operator $\mathcal{C}_{\text{sum}}^\tau(\cdot) : \mathbb{R}^{I\times d} \to \mathbb{R}^{I\times d}$ with a fixed iteration number τ, and outline its detailed steps as the following Algorithm 8.

Algorithm 8: Sum Consensus – $\mathcal{C}_{\text{sum}}^{\tau}(\cdot)$

Data: Each UAV receives data $\mathbf{u}_i \in \mathbb{R}^d$, specifies the row-stochastic weights w_{ij}'s, and initializes $\mathbf{x}_i^0 = \mathbf{u}_i$, $\boldsymbol{\phi}_i^0 = \mathbf{e}_i$. Set the maximum iteration number τ, let $k = 0$.

while the iteration index $k \leq \tau$ **do**

Each UAV $i \in \mathcal{I}$ **simultaneously** does

(S.1) Collect the information $\mathbf{x}_j^k$ and $\boldsymbol{\phi}_j^k$ from the in-neighbors $j \in \mathcal{N}_{i,+}$;

(S.2) Update the local states $\mathbf{x}_i^{k+1}$ and $\boldsymbol{\phi}_i^{k+1}$ as (6.13);

(S.3) Send the updated states to the out-neighbors;

(S.4) Let $k \leftarrow k + 1$ and continue.

end

Inherited from the convergence result as shown in Theorem 2.4.1 in Section 2.4, we state the convergence of the sum consensus operator $\mathcal{C}_{\text{sum}}^{\tau}(\cdot)$ as follows.

Theorem 6.1.1. *Suppose that the time-invariant directed graph of the multi-UAV network satisfies Assumption 1.4.7 and the compliant weight matrix W satisfies Assumptions 1.4.9 and 1.4.10, for any given set of initial states $X^0 := [\mathbf{x}_1^0, \mathbf{x}_2^0, \cdots, \mathbf{x}_I^0]^\top \in \mathbb{R}^{I \times d}$, one can have*

$$\lim_{\tau \to \infty} \mathcal{C}_{\text{sum}}^{\tau}(X^0) = \mathbf{1}_I \mathbf{1}_I^\top X^0. \tag{6.14}$$

With the help of the sum consensus operator as well as its convergence as shown in the above Theorem 6.1.1, we are now ready to develop the complete distributed data fusion algorithm to seek the first generalized eigenvector $\boldsymbol{\alpha}^\star$ for the matrix pair (R, D) as shown in (6.9). Inspired by the well-known Power Method [147] (also known as Power Iteration) which is designed to solve a standard eigenvector/eigenvalue problem, we compute the generalized eigenvector by the following distributed scheme.

Let us first give a review on the centralized Power Method. Consider a diagonalizable matrix $A \in \mathbb{R}^{I \times I}$, starting with an initial $\mathbf{v}^0 \in \mathbb{R}^I$, then the sequence $\{\mathbf{v}^k\}_{k=0}^{\infty}$ generated by

$$\mathbf{v}^{k+1} = A\mathbf{v}^k, \tag{6.15}$$

will converge to a eigenvector associated with the eigenvalue of the largest magnitude.

Remark 6.1.1. *Note that the power iteration as defined in (6.15) does not normalize at each iteration. As an estimate of the eigenvector, we are only interested in the direction, rather than the magnitude of each* $\mathbf{v}^k$*, and thus there is no need to normalize it in theory. In practice, however,* $\mathbf{v}^k$ *may either underflow when* $\|A\| < 1$ *or overflow when* $\|A\| > 1$*. In our distributed algorithm, we will introduce an additional normalization step.*

We recall that the generalized eigenvector problem to be solved follows $R\mathbf{v} = \lambda D\mathbf{v}$, and the matrix pair (R, D) in a compact form reads,

$$R = M(\mathbf{I}_d - \frac{1}{d} \cdot \mathbf{1}_d\mathbf{1}_d^\top)M^\top \quad \text{and} \quad D \ = \mathcal{D}(R), \tag{6.16}$$

where $\mathcal{D}(\cdot) : \mathbb{R}^{I\times I} \to \mathbb{R}^{I\times I}$ takes the diagonal matrix. Applying the standard power method (6.15) to the generalized eigenvector problem leads to an iterative procedure,

$$D\mathbf{v}^{k+1} = R\mathbf{v}^k. \tag{6.17}$$

Note that the above procedure will require a solution to the system of linear equations. Owing to the specific structure of matrices R and D, we can further rewrite (6.17) as

$$\mathbf{v}^{k+1} = D^{-1}R\mathbf{v}^k = D^{-1}M(\mathbf{I}_d - \frac{1}{d} \cdot \mathbf{1}_d\mathbf{1}_d^\top)M^\top\mathbf{v}^k, \tag{6.18}$$

and it can be validated by the following lemma.

Lemma 6.1.1. *Suppose that the matrix pair* (R, D) *is defined as (6.16) and the matrix* M *is defined in Section 6.1.1, then the following statements hold,*

i) R *is positive semi-definite;*

ii) D *is positive definite with probability one;*

iii) $D^{-1}R$ *is positive semi-definite.*

Proof. i) Given that $R = M\Big(\mathbf{I}_d - (1/d)\cdot \mathbf{1}_d\mathbf{1}_d^\top\Big)M^\top$, it is sufficient to prove that $\mathbf{I}_d - (1/d)\cdot \mathbf{1}_d\mathbf{1}_d^\top$ is positive semi-definite. In fact, for $\forall \boldsymbol{\zeta} \in \mathbb{R}^d$, we can have

$$\boldsymbol{\zeta}^\top(\mathbf{I}_d - \frac{1}{d}\cdot \mathbf{1}_d\mathbf{1}_d^\top)\boldsymbol{\zeta} = \|\boldsymbol{\zeta}\|^2 - \frac{1}{d}\cdot(\mathbf{1}_d^\top\boldsymbol{\zeta})^2 \geq 0. \tag{6.19}$$

Note that the last inequality is due to the Cauchy-Schwartz inequality. Therefore, R is positive semi-definite.

ii) Since the matrix $R = [r_{ij}]_{i,j=1}^I$ is positive semi-definite, its diagonal element has

$$r_{ii} \geq 0,\ i \in \mathcal{I}. \tag{6.20}$$

Recall that D is the diagonal matrix of R, i.e., $D = \mathcal{D}(R)$, we can have that D is also positive semi-definite. Moreover, note that $r_{ii} = \|\mathbf{m}_i\|^2 - d\bar{m}_i^2$ and $\mathbf{m}_i = \boldsymbol{\Gamma}_i\mathbf{s}_0 + \mathbf{b}_i + \mathbf{n}_i$ with $\mathbf{n}_i$ follows Gaussian distribution, then $r_{ii} \neq 0$ holds with probability one. Hence, D is positive definite with probability one.

iii) Since D is positive definite, let us denote $D^{-1/2} = \text{diag}\{1/\sqrt{r_{11}}, 1/\sqrt{r_{22}}, \cdots, 1/\sqrt{r_{II}}\}$. Then, one can have that $D^{-1}R$ is similar to $D^{-1/2}RD^{-1/2}$ by

$$D^{-1}R = D^{-1/2}(D^{-1/2}RD^{-1/2})D^{1/2}. \tag{6.21}$$

Since $D^{-1/2}RD^{-1/2}$ is positive semi-definite, so is $D^{-1}R$. □

Leveraging the idea presented in [148], we now present our distributed scheme to perform the iteration (6.17). Suppose that each component v_i^k in $\mathbf{v}^k = [v_1^k, v_2^k, \cdots, v_I^k]^\top$ is updated by the i-th UAV, we can have the local update as

$$v_i^{k+1} = \frac{1}{r_{ii}}\mathbf{m}_i^\top(\mathbf{I}_d - \frac{1}{d}\cdot \mathbf{1}_d\mathbf{1}_d^\top)\cdot\sum_{i=1}^I v_i^k\mathbf{m}_i. \tag{6.22}$$

Since $\mathbf{m}_i$ represents the local measurement of the i-th UAV, we remark that in (6.22) every piece of information except the summation term can be locally obtained. Subsequently, the summation term can be viewed as one component of the output from a sum consensus op-

erator $\mathcal{C}^{\tau}_{\text{sum}}(\cdot)$ defined in Algorithm 8. Thus, we obtain the fully distributed implementation of (6.22), as outlined in Algorithm 9.

Algorithm 9: Distributed Data Fusion

Data: Each UAV i obtains $\mathbf{m}_i \in \mathbb{R}^d$ and initializes $v_i^0 = 1$. Let $k = 0$.

while the termination criterion is NOT satisfied **do**

Each UAV $i \in \mathcal{I}$ **simultaneously** does

`(S.1)` Locally compute $v_i^k \mathbf{m}_i$ and apply $\mathcal{C}^k_{\text{sum}}(\cdot)$ with the input $v_i^k \mathbf{m}_i$, obtaining the output, denoted as $\boldsymbol{\xi}_i^k$;

`(S.2)` Update the local state by

$$v_i^{k+1} \leftarrow \frac{1}{r_{ii}} \mathbf{m}_i^{\top} (\mathbf{I}_d - \frac{1}{d} \cdot \mathbf{1}_d \mathbf{1}_d^{\top}) \boldsymbol{\xi}_i^k; \tag{6.23}$$

`(S.3)` Apply $\mathcal{C}^k_{\text{sum}}(\cdot)$ with the input v_i^{k+1}, obtaining the output η_i^{k+1};

`(S.4)` Update the local states by

$$v_i^{k+1} \leftarrow \frac{v_i^{k+1}}{\eta_i^{k+1}}; \tag{6.24}$$

`(S.5)` Let $k \leftarrow k + 1$ and continue.

end

`(S.6)` Let $\beta_i^{\star} = \bar{m}_i$ and $\alpha_i^{\star} = v_i^{k+1}$, and compute $\mathbf{s}_i = \alpha_i^{\star}(\mathbf{m}_i - \beta_i^{\star} \mathbf{1}_d)$;

`(S.7)` Apply $\mathcal{C}^k_{\text{sum}}(\cdot)$ with the input $\mathbf{s}_i$, obtaining the output $\mathbf{s}^{\star}$.

Note that steps `(S.3)` and `(S.4)` in Algorithm 9 accomplish the normalization of $\mathbf{v}^k$ at each iteration. Recall that, in Section 6.1.2, to recover the true signal by the summation (6.10), we restrict the re-scaling coefficients to satisfy $\mathbf{1}_I^{\top} \boldsymbol{\alpha} = 1$. This is ensured by

$$v_i^k \leftarrow \frac{v_i^k}{\sum_{i=1}^I v_i^k}, \quad i = 1, 2, \cdots, I, \tag{6.25}$$

and the summation term in (6.25) can be also distributively obtained by another sum consensus operator $\mathcal{C}^{\tau}_{\text{sum}}(\cdot) : \mathbb{R} \to \mathbb{R}$.

Finally, we state the convergence result of Algorithm 9 as follows.

Theorem 6.1.2. *Suppose that the time-invariant directed graph of the multi-UAV network satisfies Assumption 1.4.7 and the measurements are defined in (6.2), then the sequence* $\{\mathbf{v}^k\}_{k\in\mathbb{N}_+}$ *generated by Algorithm 9 satisfies,*

$$\lim_{k\to\infty} \operatorname{dist}\Big(\operatorname{span}\{\mathbf{v}^{\mathrm{k}}\},\, \operatorname{span}\{\mathbf{u}_1\}\Big) = 0, \tag{6.26}$$

where $\mathbf{u}_1 \in \mathbb{R}^I$ *is the first generalized eigenvector of the matrix pair* (R, D)*. Thus,* $\mathbf{s}^\star$ *is an optimal recovery of the measurement, in the sense of MSNR.*

Proof. To prove the theorem, let us first show the following statement: the largest generalized eigenvalue λ_1 of the matrix pair (R, D) must have $\lambda_1 \geq 1$. In fact, the largest generalized eigenvalue of the matrix pair (R, D) can be calculated by maximizing the following Rayleigh Quotient,

$$\lambda_1 = \max_{\boldsymbol{\zeta}} \frac{\boldsymbol{\zeta}^\top R\boldsymbol{\zeta}}{\boldsymbol{\zeta}^\top D\boldsymbol{\zeta}}. \tag{6.27}$$

By Lemma 6.1.1, we have proved that R is positive semi-definite and D is positive definite with probability one, i.e., for $\forall \boldsymbol{\zeta} \neq 0$, $\boldsymbol{\zeta}^\top R\boldsymbol{\zeta} \geq 0$ and $\boldsymbol{\zeta}^\top D\boldsymbol{\zeta} > 0$. Thus, to prove $\lambda_1 \geq 1$, it is sufficient to show that there exists some $\boldsymbol{\zeta}$ such that

$$\boldsymbol{\zeta}^\top R\boldsymbol{\zeta} - \boldsymbol{\zeta}^\top D\boldsymbol{\zeta} \geq 0. \tag{6.28}$$

Since the matrix D takes the diagonal matrix of R, we can have that the diagonal elements of $R - D$ are all zeros. Suppose that the eigenvalues of $R - D$ are $\{\gamma_1, \gamma_2, \cdots, \gamma_I\}$, then

$$\sum_{i=1}^{I} \gamma_i = \operatorname{trace}(R - D) = 0, \tag{6.29}$$

where $\operatorname{trace}(\cdot) : \mathbb{R}^{I\times I} \to \mathbb{R}$ denotes the trace of the matrix. Therefore, as long as $R - D$ is not a zero matrix, these must exist a strictly positive eigenvalue γ_+, and let us denote the corresponding eigenvector as $\boldsymbol{\zeta}_+$. Now, since $\boldsymbol{\zeta}_+^\top(R - D)\boldsymbol{\zeta}_+ = \gamma_+\|\boldsymbol{\zeta}_+\|^2 \geq 0$, then the above statement is proved.

According to the sum consensus operator $\mathcal{C}^{\tau}_{\text{sum}}(\cdot)$ defined by the iteration (6.13), let us define a transfer matrix $C_\tau \in \mathbb{R}^{I\times I}$ to represent the operator, i.e.,

$$X^{\tau} = \mathcal{C}^{\tau}_{\text{sum}}(X^0) = C_\tau X^0, \tag{6.30}$$

where

$$C_\tau = (W)^{\tau} + \sum_{t=0}^{\tau-1} (W)^{\tau-t}\Big(\mathcal{D}^{-1}\big((W)^{t+1}\big) - \mathcal{D}^{-1}\big((W)^{t}\big)\Big). \tag{6.31}$$

By Theorem 6.1.1, we know that $\lim_{\tau\to\infty}\|C_\tau - \mathbf{1}_I\mathbf{1}_I^\top\| = 0$.

Now, based on the step `(S.2)` in Algorithm 9, each local update can be rewritten as

$$v_i^{k+1} = \frac{1}{r_{ii}}\mathbf{m}_i^\top(\mathbf{I}_d - \frac{1}{d}\cdot\mathbf{1}_d\mathbf{1}_d^\top)\cdot[C_k\mathcal{D}_{\text{m}}(\mathbf{v}^k)M]_i^\top, \tag{6.32}$$

where $\mathcal{D}_{\text{m}}(\cdot):\mathbb{R}^I\to\mathbb{R}^{I\times I}$ maps the vector into a diagonal matrix and $[\,\cdot\,]_i:\mathbb{R}^{I\times I}\to\mathbb{R}^I$ takes the i-th row of the matrix. Alternatively, in a compact form, we can have

$$\mathbf{v}^{k+1} = \mathcal{D}_{\text{v}}\Big(D^{-1}M(\mathbf{I}_d - \frac{1}{d}\cdot\mathbf{1}_d\mathbf{1}_d^\top)\big(C_k\mathcal{D}_{\text{m}}(\mathbf{v}^k)M\big)^\top\Big), \tag{6.33}$$

with $\mathcal{D}_{\text{v}}(\cdot):\mathbb{R}^{I\times I}\to\mathbb{R}^I$ maps the diagonal entries of the matrix into a vector. Moreover, let us denote $A = D^{-1}R$ and recall that $\mathcal{D}_{\text{m}}(\mathbf{v}^k)$ is symmetric, the update can be further written as

$$\mathbf{v}^{k+1} = \mathcal{D}_{\text{v}}\left(A\mathcal{D}_{\text{m}}(\mathbf{v}^k)C_k^\top\right). \tag{6.34}$$

Given that the matrix C_k has $\lim_{k\to\infty}\|C_k - \mathbf{1}_I\mathbf{1}_I^\top\| = 0$, we can next explore the consensus error by rewriting (6.34) as follows

$$\begin{aligned}
\mathbf{v}^{k+1} &= \mathcal{D}_{\text{v}}\left(A\mathcal{D}_{\text{m}}(\mathbf{v}^k)(C_k^\top - \mathbf{1}_I\mathbf{1}_I^\top + \mathbf{1}_I\mathbf{1}_I^\top)\right)\\
&= A\mathbf{v}^k + \mathcal{D}_{\text{v}}\left(A\mathcal{D}_{\text{m}}(\mathbf{v}^k)(C_k^\top - \mathbf{1}_I\mathbf{1}_I^\top)\right)\\
&= (A)^{k+1}\mathbf{v}^0 + \sum_{t=0}^{k}(A)^{k-t}\mathcal{D}_{\text{v}}\left(A\mathcal{D}_{\text{m}}(\mathbf{v}^t)(C_t^\top - \mathbf{1}_I\mathbf{1}_I^\top)\right).
\end{aligned} \tag{6.35}$$

Note that here we expect the vector sequence $\{\mathbf{v}_k\}_{k\in\mathbb{N}_+}$ generated by (6.35) converges to the first eigenvector of A, i.e., the first generalized eigenvector of (R, D). According to the convergence result of centralized power method [147], the term $(A)^{k+1}\mathbf{v}^0$ converges to the expected eigenvector. Now we will only need to show the summation term also converges to the eigenvector.

Notice that $A = D^{-1}R$ can be diagonalized into $U^{-1}AU = \text{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_I\}$ with $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_I$ and $U := [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_I] \in \mathbb{R}^{I\times I}$. Furthermore, the group of eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_I\}$ forms a basis of $\mathbb{R}^I$. Next, let us define

$$\boldsymbol{\epsilon}^k = \sum_{t=0}^{k}(A)^{k-t}\mathcal{D}_{\text{v}}\left(A\mathcal{D}_{\text{m}}(\mathbf{v}^t)(C_t^\top - \mathbf{1}_I\mathbf{1}_I^\top)\right), \tag{6.36}$$

and represent the consensus error term in (6.36) by the basis, i.e.,

$$\mathcal{D}_{\text{v}}\left(A\mathcal{D}_{\text{m}}(\mathbf{v}^t)(C_t^\top - \mathbf{1}_I\mathbf{1}_I^\top)\right) = \sum_{i=1}^{I} a_i^t\mathbf{u}_i. \tag{6.37}$$

Due to the fact that $\lim_{k\to\infty}\|C_k - \mathbf{1}_I\mathbf{1}_I^\top\| = 0$ and $\mathcal{D}_{\text{m}}(\mathbf{v}^t)$ is always bounded, the the sequence $\{a_i^t\}_{t\in\mathbb{N}_+}$ satisfies $\lim_{t\to\infty} a_i^t = 0$. Therefore, we plug (6.37) into (6.36), yielding

$$\begin{aligned}
\boldsymbol{\epsilon}^k &= \sum_{i=1}^{I}\sum_{t=0}^{k} a_i^t(\lambda_i)^{k-t}\mathbf{u}_i \\
&= (\lambda_1)^k\sum_{i=1}^{I}\sum_{t=0}^{k} a_i^t(\lambda_1)^{-t}(\lambda_i/\lambda_1)^{k-t}\mathbf{u}_i \\
&= (\lambda_1)^k\left(\sum_{t=0}^{k} a_1^t(\lambda_1)^{-t}\mathbf{u}_1 + \sum_{i=2}^{I}\sum_{t=0}^{k} a_i^t(\lambda_1)^{-t}(\lambda_i/\lambda_1)^{k-t}\mathbf{u}_i\right).
\end{aligned} \tag{6.38}$$

Since $\lambda_1 \geq 1$, we have $\lim_{t\to\infty} a_i^t(\lambda_1)^{-t} = 0$. Recall the crucial Lemma 1.4.2 in Chapter 1, together with the fact $\lambda_i/\lambda_1 < 1$ for $\forall i = 2, 3, \cdots, I$, we can have

$$\lim_{k\to\infty}\sum_{t=0}^{k} a_i^t(\lambda_1)^{-t}(\lambda_i/\lambda_1)^{k-t} = 0, \quad i = 2, 3, \cdots, I. \tag{6.39}$$

Hence, it holds that $\lim_{k\to\infty}\text{dist}(\text{span}\{\boldsymbol{\epsilon}^k\}, \text{span}\{\mathbf{u}_1\}) = 0$, and thus Theorem 6.1.2 is proved.

□

6.1.4 Numerical Experiment

In this subsection, we evaluate our distributed data fusion framework via a set of simulations by considering the potential rural health monitoring application. In our simulation, we adopt the noisy measurement model in Section 6.1.1. A simulated biological electrocardiogram (ECG) signal is assumed to be sensed by the multi-UAV network. The target ECG signal is generated via a simulator provided in [149]. The heart rate is assumed to be 105 beats per minute (bpm) and the desired peak voltage is 1.2 millivolts (mV). Fig. 6.1 shows the simulated ECG signal within a 10-second time interval.

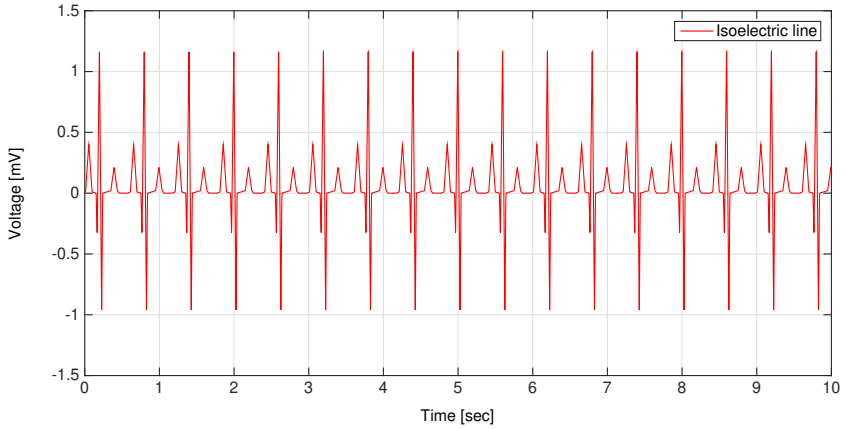


Figure 6.1. Artificial ECG signal

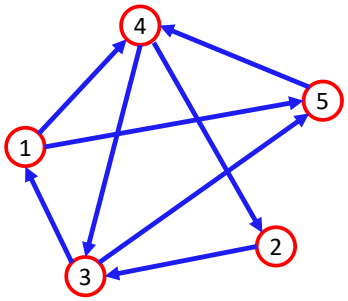


Figure 6.2. Topology of a 5-UAV network

We first consider a small-size multi-UAV network, composed of 5 UAVs, for such rural health monitoring mission. The communication channels among the multi-UAV network are specified as a connected time-invariant directed graph as shown in Fig. 6.2. The received

noisy measurement $\mathbf{m}_i$ by each UAV is randomly generated by the measurement model as shown in (6.2). Fig. 6.3 plots those noisy measurements; while the red lines represent the true signal $\mathbf{s}_0$, the black lines are measurements by each UAV. As demonstrated in Fig. 6.3, every sensed signal is ill-polluted to some extent. While the first three are possibly imperfect with being differently scaled and biased, the last two could be completely useless as they lose the main features of the signal. Therefore, it is reasonable to believe that each sensed signal is not reliable and thus should be fused to obtain a more accurate result. We demonstrate the performance of the distributed data fusion algorithm in the next subsection.

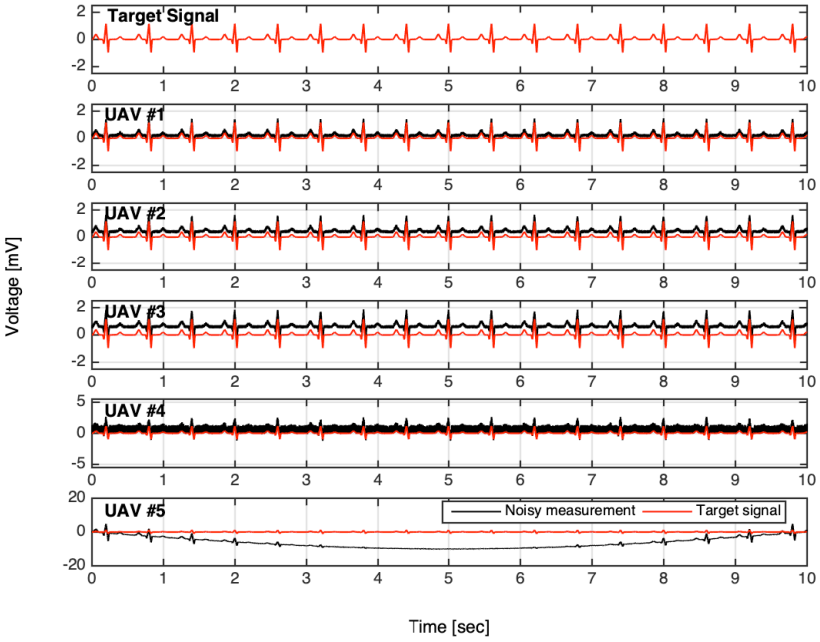


Figure 6.3. Measurements of the signal within the network

In Fig. 6.4, we show the results obtained by both a pure averaging scheme (the top plot) and a centralized optimal averaging scheme (the bottom plot). It can be seen that the result from pure averaging is heavily biased, due to the dominating magnitude of the worst measurement from the fifth UAV. On the contrary, the centralized optimal averaging scheme offers the much better estimates. In fact, as reflected in the computed optimal re-scaling coefficients, i.e., $\boldsymbol{\alpha}^\star = [0.2972, 0.2966, 0.2991, 0.1025, 0.0047]^\top$, the optimal averaging gives smaller weights to more ill-polluted measurements.

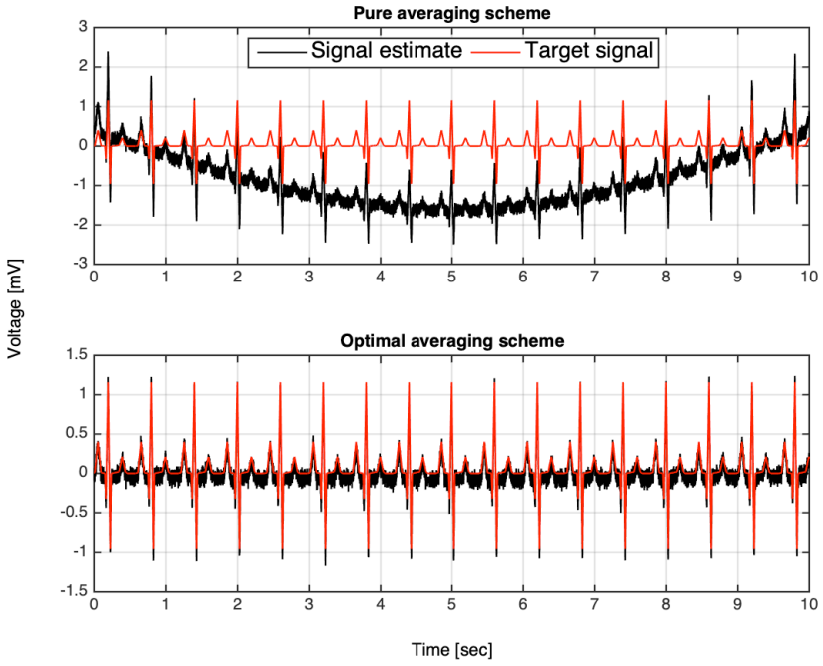


Figure 6.4. Comparison between pure averaging and optimal averaging

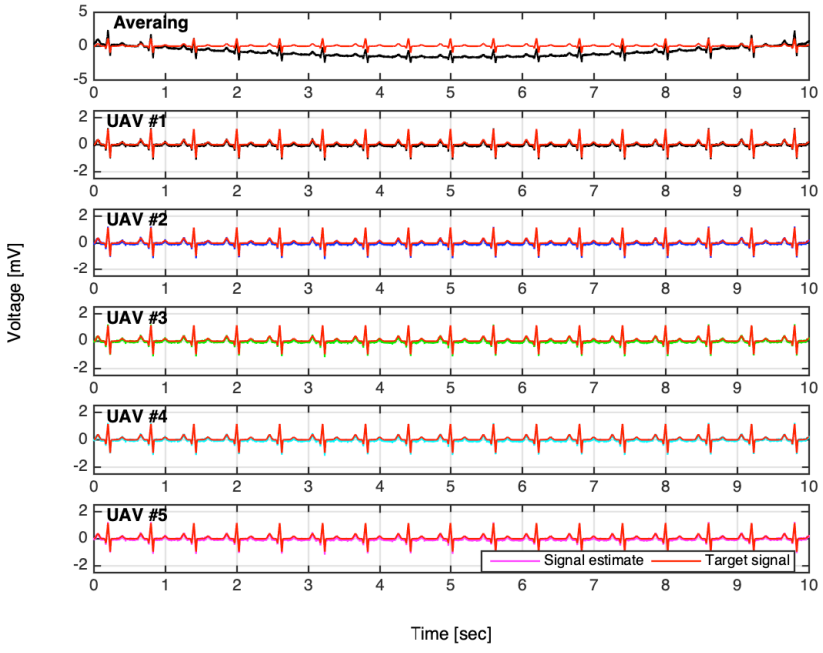


Figure 6.5. Performance of the distributed data fusion (5 UAVs)

We now test the proposed distributed approach (Algorithm 9) equipped with the sum consensus scheme (Algorithm 8). Fig. 6.5 illustrates the results obtained after the 23-rd iteration and provides a comparison between the estimates locally obtained by each UAV and the one obtained by pure averaging. As shown in Fig. 6.5, each UAV can obtain an estimate with high accuracy. In this sense, any UAV is able to send a reliable measurement back to the medical center. Given that only a few but not all of the UAVs are capable of sending information back, our framework offers flexibility and efficiency.

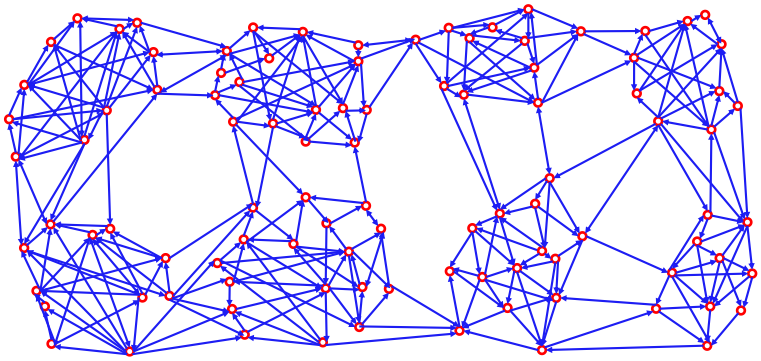


Figure 6.6. Topology of an 100-UAV network

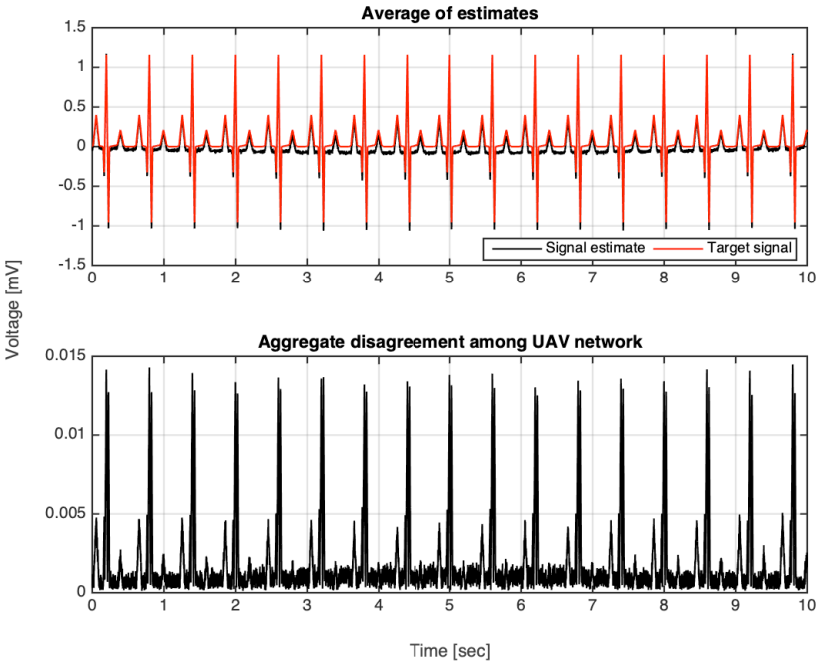


Figure 6.7. Performance of the distributed data fusion (100 UAVs)

We further evaluate the proposed algorithm by simulating a large-size UAV network composed of 100 UAVs. The network topology is shown as Fig. 6.6. Suppose that the true signal recovery generated by each individual UAV i is denoted as $\mathbf{s}_i^\star$, we evaluate the distributed sensor fusion algorithm in two aspects. First, we take the average of all estimates, i.e., $\bar{\mathbf{s}}^\star = (1/I) \cdot \sum_{i=1}^{I} \mathbf{s}_i^\star$, and use the average $\bar{\mathbf{s}}^\star$ to demonstrate a fused result. Second, we measure the disagreement among all UAVs within the network by the consensus error $\boldsymbol{J} = (1/I) \cdot \sum_{i=1}^{I} |\mathbf{s}_i^\star - \bar{\mathbf{s}}^\star|$, where $|\cdot| : \mathbb{R}^d \to \mathbb{R}^d$ takes the absolute values element-wise. In Fig. 6.7, the figure on the top plots the average of estimates $\bar{\mathbf{s}}^\star$. It can be seen that $\bar{\mathbf{s}}^\star$ recovers the true signal with high accuracy. Meanwhile, the figure on the bottom demonstrates the consensus error. By combining these two figures together, it is verified that the network of UAVs can reach a consensus at which each UAV provides a fused result with high accuracy.

6.2 Application II: Distributed Source Tracking in Unknown Environments

In the second application, we are interested in the distributed source tracking problem [150], [151], i.e., locating one or several positions of the sources, associated with measurements maxima, in a possibly unknown and noisy environment. More specifically, we employ a multi-UAV system and expect them to cooperatively locate as many local maxima sources as possible, by leveraging the communications among different UAVs. In addition, we consider that the environment is not only unknown but also changing dynamically as the multiple UAVs acquire knowledge from it. In this situation, the team of UAVs needs to track the moving sources in real-time. We remark that these two settings, i.e., the multi-UAV system and dynamical environment, make our problem challenging to solve.

6.2.1 Source Tracking with the Multi-UAV System

Let us consider a bounded and obstacle-free environment, in which sources of interest are present. In particular, we specify the considered environment by a set of points $\mathcal{S}$ with each element $\mathbf{s} \in \mathcal{S}$ representing the position of the point. Since the environment has been assumed to be bounded, it is easy to see that the set $\mathcal{S}$ is finite. We denote N the number of points in the set, i.e., $N = |\mathcal{S}|$. For each point $\mathbf{s}$ in $\mathcal{S}$, there exists a real-

valued function $\phi_k(\cdot) : \mathcal{S} \to \mathbb{R}_+$ that maps the point's positional information $\mathbf{s}$ to a positive quantity $\phi_k(\mathbf{s})$ indicating the level of the source at the time-step k. Naturally, in order to locate the sources, our objective is to deploy the multiple UAVs to the points with the highest quantities $\phi_k(\mathbf{s})$. More specifically, let us employ a team of I UAVs which are capable of moving among $\mathcal{S}$ and communicating with other connected UAVs, and expect them to track as many sources as possible. That is, at each time-step k, the multiple UAVs aims at seeking their best positions $\mathbf{p}_k^\star[i] \in \mathcal{S},\ i \in \mathcal{I} = \{1, 2, \cdots, I\}$ by cooperatively solving the following maximization problem,

$$\max_{\mathbf{p}[i]\in\mathcal{S},\, i\in\mathcal{I}} \quad F_k(\mathbf{p}[1], \mathbf{p}[2], \cdots, \mathbf{p}[I]) = \sum_{\mathbf{s}\in\cup_{i=1}^{I}\mathbf{p}[i]} \phi_k(\mathbf{s}). \tag{6.40}$$

Note that the objective function $F_k(\cdot) : \mathcal{S}^I \to \mathbb{R}_+$ maps the UAVs' positions $\mathbf{p}[i]$'s to a positive scalar that sums all distinct measured quantities. Here, let us assume that the maximizer $\Big(\mathbf{p}_k^\star[1], \mathbf{p}_k^\star[2], \cdots, \mathbf{p}_k^\star[I]\Big)$ of problem (6.40) is unique at each time-step k and express it as a compact form $\mathbf{p}_k^\star = \Big[\mathbf{p}_k^\star[i]\Big]_{i\in\mathcal{I}} \in \mathcal{S}^I$.

It should be remarked that, since the set $\mathcal{S}$ is finite, the above maximization problem can be naively solved by assigning the i-th UAV to the point $\mathbf{p}[i]$ which has the i-th largest quantity $\phi_k\Big(\mathbf{p}[i]\Big)$. However, such a naive scheme inherently assumes each UAV to be aware of its exclusive global ID which is a restrictive requirement in the distributed setting [152]. As an alternative way to solve the optimization problem (6.40), we shall remark that the problem can be viewed as a special case of the monotone submodular maximization, and thus can be solved by the distributed algorithm proposed in our previous work [153]. The key idea of this algorithm is to find the equilibrium solution, and interestingly, it can be verified that the problem (6.40) has a unique equilibrium which is coincident with the optimal solution.

Notice that the problem (6.40) considered in the above context is somewhat trivial, since it implicitly assumes that each UAV perfectly knows the state $\phi_k(\mathbf{s})$ of the entire environment at each time-step k. This is unrealistic for the real-world applications. To respond, we next let the team of UAVs cooperatively estimate the environment based on the local noisy measurements, and in the following, we first introduce the dynamics of the environment states as well as the measurement model of the UAVs.

Suppose that the vector $\boldsymbol{\phi}_k \in \mathbb{R}_+^N$ stacks each individual state $\phi_k(\mathbf{s})$ for all points $\mathbf{s}$ in the environment $\mathcal{S}$. We consider the following linear time-varying (LTV) model for the environment state, i.e.,

$$\boldsymbol{\phi}_{k+1} = A_{k+1}\boldsymbol{\phi}_k, \tag{6.41}$$

where $A_k \in \mathbb{R}^{N\times N}$ denotes the state transition matrix. In order to ensure that the above maximization problem (6.40) is well-defined, it is required to guarantee that the state $\boldsymbol{\phi}_k$ is always bounded and also will not vanish to zero as the time-step k increases. More precisely, we use the following assumption to constrain the behavior of the state dynamics.

Assumption 6.2.1. *For the LTV model (6.41), there exist uniform lower and upper bounds $0 < \underline{\alpha} \leq \bar{\alpha} < \infty$ such that, for $\forall k \geq t > 0$,*

$$\underline{\alpha} \cdot \mathbf{I}_N \leq A[k:t]^\top A[k:t] \leq \bar{\alpha} \cdot \mathbf{I}_N, \tag{6.42}$$

where the state propagation matrix $A[k:t] \in \mathbb{R}^{N\times N}$ is written as

$$A[k:t] = A_k A_{k-1} \cdots A_t. \tag{6.43}$$

Remark 6.2.1. *Note that the above Assumption 6.2.1 is reasonably required to ensure that the maximum components of $\boldsymbol{\phi}_k$ are always recognizable for the multi-UAV system. Moreover, this assumption also implies the invertibility of the matrices A_k's. In fact, in the sampled-data system (one of the mostly studied discrete-time systems), the matrix A_k is naturally invertible since it is often obtained by discretization of the continuous-time system [154]. Such an assumption has been commonly made in various research studying the state estimation problems, see e.g., [154]–[157].*

In addition, we consider the following linear stochastic measurement model for each UAV i,

$$\mathbf{z}_k^i = H^i\Big(\mathbf{p}_k[i]\Big)\boldsymbol{\phi}_k + \mathbf{n}_k^i. \tag{6.44}$$

where $\mathbf{z}_k^i \in \mathbb{R}^m$ represents the measurement obtained by the UAV at the time-step k[1]; $H^i\Big(\mathbf{p}_k[i]\Big) \in \mathbb{R}^{m\times N}$ denotes the measurement matrix depending on the UAV's position $\mathbf{p}_k[i]$; and $\mathbf{n}_k^i \in \mathbb{R}^m$ is corresponding to the measurement noise satisfying the following assumption.

Assumption 6.2.2. *It is assumed that the measurement noise $\mathbf{n}_k^i$ follows the independent and identically distributed (i.i.d.) Gaussian for each individual UAV, with zero-mean and covariance matrix $V^i = v^i \cdot \mathbf{I}_m$. In addition, there exist lower and upper bounds $0 < \underline{v} \leq \bar{v} < \infty$ such that*

$$\underline{v} \leq v^i \leq \bar{v},\ \forall i \in \mathcal{I}. \tag{6.45}$$

Remark 6.2.2. *We shall remark that the measurement matrix $H^i\Big(\mathbf{p}_k[i]\Big)$ is not specified in the above model (6.44). In fact, it can be defined by various means based on the UAV's position. One of the simplest way is to let $H^i\Big(\mathbf{p}_k[i]\Big) = \mathbf{e}_l^\top$ where $\mathbf{e}_l \in \mathbb{R}^N$ is the unit vector and $l \in \{1, 2, \cdots, N\}$ denotes the index of the position $\mathbf{p}_k[i]$ in the environment $\mathcal{S}$. This means that the UAV only measures the quantity at the point where it currently is. Such a choice of $H^i\Big(\mathbf{p}_k[i]\Big)$ is actually adopted in [158] as the so-called point-wise sensing model. Besides, some other specifications of the measurement matrix are also used in the existing works. For instance, a circular sensing area with radius r_i is applied in [159], which implies that,*

$$H^i\Big(\mathbf{p}_k[i]\Big) = [\mathbf{e}_l]_{l\in\mathcal{C}_k^i}^\top, \tag{6.46}$$

where the set $\mathcal{C}_k^i := \{l \mid \|\mathbf{s}_l - \mathbf{p}_k[i]\| \leq r^i\}$ includes the indices of all points $\mathbf{s}_l$ that fall into the disk which is centered at $\mathbf{p}_k[i]$ and has radius r^i.

Based on the measurement model (6.44), one should notice that the true value of $\boldsymbol{\phi}_k$ can be estimated by many techniques, such as the least-square, the classical Kalman filter, to name a few, when some mild conditions on the measurement matrices are satisfied. Therefore, the problem of distributed source tracking with an unknown environment can be

[1]↑For simplicity, we assume that each UAV's measurement has the same dimension m; this can be easily relaxed to a general case.

addressed by a simple approach which contains the following two phases separately: i) let the team of UAVs move around the environment and obtain an accurate enough estimation of the state; and ii) specify the UAVs' target positions by solving the maximization problem (6.40) based on the estimated states. However, this is essentially an off-line approach, since the UAVs do not have specific targets when estimating the environment in the phase i) and the phase ii) cannot be started until an accurate enough estimate is obtained. Motivated by this, in the next section, we aim to integrate the above two phases together and propose an adaptive on-line framework. That is, the UAVs recursively update their target positions; meanwhile, measure and estimate the unknown environment, until the steady state is reached in which the team of I UAVs manages to tracking the top I moving sources.

6.2.2 An Adaptive On-line Framework

Let us begin by rewriting the measurement model (6.44) into the following compact form,

$$\mathbf{z}_k = H_k \boldsymbol{\phi}_k + \mathbf{n}_k. \tag{6.47}$$

Here, $\mathbf{z}_k = [(\mathbf{z}_k^1)^\top, (\mathbf{z}_k^2)^\top, \cdots, (\mathbf{z}_k^I)^\top]^\top \in \mathbb{R}^M$ is the measurement obtained by all UAVs with dimension $M = mI$; $H_k = \left[H^1(\mathbf{p}_k[1])^\top, H^2(\mathbf{p}_k[2])^\top, \cdots, H^I(\mathbf{p}_k[I])^\top\right]^\top \in \mathbb{R}^{M\times N}$ stacks all local measurement matrices as a collective global one[2]; and $\mathbf{n}_k = [(\mathbf{n}_k^1)^\top, (\mathbf{n}_k^2)^\top, \cdots, (\mathbf{n}_k^I)^\top]^\top \in \mathbb{R}^M$ denotes the Gaussian noise with zero-mean and covariance expressed as

$$V := \mathrm{diag}\{V^1, V^2, \cdots, V^I\} \in \mathbb{R}^{M\times M}. \tag{6.48}$$

Subsequently, the centralized Kalman filter for estimating the mean $\widehat{\boldsymbol{\phi}}_k \in \mathbb{R}^N$ and covariance $\Sigma_k \in \mathbb{R}^{N\times N}$ performs the following recursions,

$$\Sigma_{k+1} = A_{k+1}\Big(\Sigma_k^{-1} + Y_k\Big)^{-1} A_{k+1}^\top; \tag{6.49a}$$

$$\widehat{\boldsymbol{\phi}}_{k+1} = A_{k+1}\Big(\widehat{\boldsymbol{\phi}}_k + (\Sigma_k^{-1} + Y_k)^{-1}(\mathbf{y}_k - Y_k\widehat{\boldsymbol{\phi}}_k)\Big), \tag{6.49b}$$

[2]↑When writing H_k, with slight abuse of notation, we have absorbed the dependency on the UAVs' positions $\mathbf{p}_k[i]$'s into the index k.

where the two variables $Y_k := (H_k)^\top V^{-1} H_k \in \mathbb{R}^{N\times N}$ and $\mathbf{y}_k := (H_k)^\top V^{-1}\mathbf{z}_k \in \mathbb{R}^N$, often referred to as the new information, incorporate the measurements into the updates.

It is also worth mentioning that the Kalman filter (6.49) readily estimates the unknown environment in an on-line manner, i.e., the team of I UAVs moves to new positions, obtains the new measurements, and updates their estimations. However, we should note that the two following issues may arise: i) the statistical property of the classical Kalman filter may no longer hold due to the sequential decision process; and ii) such an on-line procedure is performed in a centralized way, since the new information Y_k and $\mathbf{y}_k$ are involved with the data obtained/maintained by all UAVs. In order to devise a distributed scheme to run the Kalman filter (6.49), many existing works, e.g., [160]–[162], leverage the special structure of the noise covariance V. Considering the diagonal structure of the matrix V, as shown in (6.48), the new information can be further expressed as

$$Y_k = \sum_{i=1}^{I} H^i(\mathbf{p}_k^i)(V^i)^{-1} H^i(\mathbf{p}_k^i)^\top; \tag{6.50a}$$

$$\mathbf{y}_k = \sum_{i=1}^{I} H^i(\mathbf{p}_k^i)(V^i)^{-1}\mathbf{z}_k^i, \tag{6.50b}$$

which means that Y_k and $\mathbf{y}_k$ can be computed by simply summing all the local information together. This motivates the development of Kalman consensus filter, in which each UAV first carries out an sum consensus procedure as introduced in Chapter 2 and also in Section 6.1.3, to fuse local information, and then performs the standard Kalman update (6.49).

Given that the distributed estimation of the unknown environment can be accomplished by the above Kalman consensus filter, our question now becomes how to integrate the estimation together with the UAVs' decision-making process. A naive idea would be using the estimated mean value $\widehat{\boldsymbol{\phi}}_k$ at each iteration k, and then solving the following maximization,

$$\mathbf{p}_k \in \underset{\mathbf{p}[i]\in\mathcal{S},\, i\in\mathcal{I}}{\arg\max} \sum_{\mathbf{s}\in\cup_{i=1}^{I}\mathbf{p}[i]} \widehat{\phi}_k(\mathbf{s}). \tag{6.51}$$

Here, we use $\widehat{\phi}_k(\mathbf{s}) \in \mathbb{R}$ to denote one component of the vector $\widehat{\boldsymbol{\phi}}_k$ which corresponds to the point $\mathbf{s}$ in the environment. It should be emphasized that such a scheme cannot guarantee

the team of UAVs to locate the sources with the highest true $\phi_k(\mathbf{s})$'s. To elaborate on this, let us consider a special case in which the environment is static, i.e., $\phi_k(\mathbf{s}) = \phi_0(\mathbf{s})$, $\forall k \in \mathbb{N}_+$. Then, an undesired but possible scenario is that the UAVs significantly underestimate the maximum value $\phi_0(\mathbf{s}^\star)$ at the initial stage, i.e., $\widehat{\phi}_0(\mathbf{s}^\star) \ll \phi_0(\mathbf{s}^\star)$, and as a result, the UAVs will never have another chance to visit the key point $\mathbf{s}^\star$. On this account, it can been seen that merely utilizing the estimated mean is insufficient to drive the team of UAVs to the desired positions. To address this, we next take advantage of both the estimated mean $\widehat{\boldsymbol{\phi}}_k$ and covariance Σ_k to develop our distributed on-line source tracking algorithm.

Based on $\widehat{\boldsymbol{\phi}}_k$ and Σ_k, let us introduce an additional variable $\boldsymbol{\mu}_k \in \mathbb{R}^N$, which we refer to as the dummy upper confidence bound (D-UCB),

$$\boldsymbol{\mu}_k := \widehat{\boldsymbol{\phi}}_k + \beta_k(\delta) \cdot \mathrm{diag}^{1/2}(\Sigma_k). \tag{6.52}$$

Note that the operator $\mathrm{diag}^{1/2}(\cdot) : \mathbb{R}^{N\times N} \to \mathbb{R}^N$ maps the square root of the matrix diagonal elements to a vector, and the parameter $\beta_k(\delta) > 0$ depending on the critical confidence level δ will be specified later on. In fact, the intuition behind this notion of D-UCB is straightforward: each $\boldsymbol{\mu}_k$ provides a probabilistic upper bound of the true value $\boldsymbol{\phi}_k$ by utilizing the current mean and covariance. Next, we formalize, with the following proposition, how the true value $\boldsymbol{\phi}_k$ is upper bounded by the D-UCB $\boldsymbol{\mu}_k$ with the probability related to δ. We postpone the proof to the next subsection.

Proposition 6.2.1. *Under Assumption 6.2.1 and 6.2.2, suppose that $\widehat{\boldsymbol{\phi}}_k$ and Σ_k are generated by the Kalman filter (6.49) with $\widehat{\boldsymbol{\phi}}_0$ and $\underline{\sigma} \cdot \mathbf{I}_N \leq \Sigma_0 \leq \bar{\sigma} \cdot \mathbf{I}_N$, then it holds that,*

$$\mathbb{P}\Big(\Big|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_0\Big| \leq \beta_k(\delta) \cdot \mathrm{diag}^{1/2}(\Sigma_k)\Big) \geq 1 - \delta, \tag{6.53}$$

where the sequence $\{\beta_k(\delta)\}_{k\in\mathbb{N}_+}$ is non-decreasing satisfying

$$\beta_k(\delta) \geq N^{3/2}c_1 + N^2 c_2 \cdot \sqrt{\log\Big(\frac{\bar{\sigma}/\underline{\sigma} + \bar{\alpha}\bar{\sigma}\cdot k/\underline{v}^2}{\delta^{2/N}}\Big)}, \tag{6.54}$$

with $c_1 = \|\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0\|/\sqrt{\underline{\sigma}}$ and $c_2 = \bar{v}^2\sqrt{\max\{2, 2/\underline{v}\}}$.

The above Proposition 6.2.1 inherently constructs a polytope centered at the state estimate $\hat{\boldsymbol{\phi}}_k$ such that the true state $\boldsymbol{\phi}_k$ falls into it with probability at least $1-\delta$. Based on the polytope, it can be seen that the D-UCB $\boldsymbol{\mu}_k$ takes the upper bounds marginally and each element $\mu_k(\mathbf{s})$ is guaranteed to have $\mu_k(\mathbf{s}) \geq \phi_k(\mathbf{s})$ with probability at least $1-\delta$. Now, we can use the defined D-UCB $\boldsymbol{\mu}_k$ to update the UAVs target positions in the on-line manner, by solving the following maximization problem:

$$\mathbf{p}_k \in \underset{\mathbf{p}[i]\in\mathcal{S},\, i\in\mathcal{I}}{\arg\max} \sum_{\mathbf{s}\in\cup_{i=1}^{I}\mathbf{p}[i]} \mu_k(\mathbf{s}). \tag{6.55}$$

We summarize our complete distributed on-line source tracking scheme as Algorithm 10 and establish its performance with the following theorem. Likewise, the proof is postponed to the next subsection.

Algorithm 10: Distributed On-line Source Tracking

Data: Each UAV i initializes its own estimates $\widehat{\boldsymbol{\phi}}_0$ and Σ_0, and computes the target position $\mathbf{p}_1^i$. Set the confidence level δ and parameters $\{\beta_k(\delta)\}_{k\in\mathbb{N}_+}$, let $k=1$.

while the termination criteria is NOT satisfied **do**

Each agent $i \in \mathcal{I}$ **simultaneously** does

(S.1) Obtain the measurement $\mathbf{z}_k^i$ based on the measurement matrix $H^i(\mathbf{p}_k^i)$;

(S.2) Collect information from neighbors, obtain mean $\widehat{\boldsymbol{\phi}}_k$ and covariance Σ_k by Kalman consensus filter (6.49);

(S.3) Compute via (6.52) the updated D-UCB $\boldsymbol{\mu}_k$ based on $\widehat{\boldsymbol{\phi}}_k$ and Σ_k;

(S.4) Assign the new target position $\mathbf{p}_{k+1}^i$ by solving (6.55).

(S.5) Let $k \leftarrow k+1$, and continue.

end

Theorem 6.2.3. *Suppose that $\{\mathbf{p}^k\}_{k\in\mathbb{N}_+}$ is the sequence generated by Algorithm 10, under the conditions in Proposition 6.2.1, then it holds that, with probability $1-\delta$, for $\forall K \in \mathbb{N}_+$,*

$$\sum_{k=1}^{K}\Big(F_k(\mathbf{p}_k^\star) - F_k(\mathbf{p}_k)\Big) \leq \mathcal{O}\Big(\sqrt{K}\log(K)\Big). \tag{6.56}$$

6.2.3 Convergence Analysis

In this subsection, we prove the convergence result of our distributed on-line source tracking algorithm, stated in both Proposition 6.2.1 and Theorem 6.2.3. In order to facilitate the following proofs, let us start with introducing several vector norms. First, associated with an arbitrary positive definite matrix $M = [m_{ij}]_{i,j=1}^N \in \mathbb{R}^{N\times N}$, we define the $\mathcal{L}_2$-based vector norm $\|\cdot\|_M : \mathbb{R}^N \to \mathbb{R}_+$ as

$$\|\mathbf{x}\|_M := \sqrt{\mathbf{x}^\top M\mathbf{x}}, \tag{6.57}$$

where $\mathbf{x} = [x_1, x_2, \cdots x_N]^\top \in \mathbb{R}^N$. Further, let us define the $\mathcal{L}_\infty$-based norm $\|\cdot\|_{\mathcal{D}_M,\infty} : \mathbb{R}^N \to \mathbb{R}_+$ associated with the diagonal matrix of the arbitrary positive definite M, i.e., $\mathcal{D}_M = \text{diag}\{m_{11}, m_{22}, \cdots, m_{NN}\} \in \mathbb{R}^{N\times N}$, as

$$\|\mathbf{x}\|_{\mathcal{D}_M,\infty} := \max_{1\le i\le N} m_{ii}\cdot|x_i|. \tag{6.58}$$

Note that the above norm $\|\cdot\|_{\mathcal{D}_M,\infty}$ is well-defined since the positive definiteness of M ensures that $m_{ii} > 0$. Similarly, we define the $\mathcal{L}_1$-based norm $\|\cdot\|_{\mathcal{D}_M,1} : \mathbb{R}^N \to \mathbb{R}_+$ as

$$\|\mathbf{x}\|_{\mathcal{D}_M,1} := \sum_{i=1}^N m_{ii}\cdot|x_i|. \tag{6.59}$$

With the vector norms introduced above, it can be immediately verified that the $\mathcal{L}_1$-based norm $\|\cdot\|_{\mathcal{D}_M,1}$ is the dual norm of the $\mathcal{L}_\infty$-based $\|\cdot\|_{\mathcal{D}_M^{-1},\infty}$ where $\mathcal{D}_M^{-1}$ takes the inverse of the matrix $\mathcal{D}_M$, and in addition, for $\forall\mathbf{x}\in\mathbb{R}^N$,

$$\|\mathbf{x}\|_{\mathcal{D}_M,\infty} \le \|\mathbf{x}\|_{\mathcal{D}_M,1} \le \sqrt{N}\cdot\|\mathbf{x}\|_{\mathcal{D}_M^2}. \tag{6.60}$$

Furthermore, we show, by the following lemma, the relationship between $\|\mathbf{x}\|_M$ and $\|\mathbf{x}\|_{\mathcal{D}_M}$.

Lemma 6.2.1. *For arbitrary positive definite $M \in \mathbb{R}^{N\times N}$, it holds that $\forall\mathbf{x}\in\mathbb{R}^N$,*

$$\|\mathbf{x}\|_M \le N\cdot\|\mathbf{x}\|_{\mathcal{D}_M}. \tag{6.61}$$

Proof. According to the above definitions, one can have

$$\begin{aligned}
\|\mathbf{x}\|_M^2 &= \sum_{i=1}^{N}\sum_{j=1}^{N} m_{ij}\cdot x_i x_j \\
&\leq \sum_{i=1}^{N} m_{ii}\cdot x_i^2 + \sum_{i=1}^{N}\sum_{j\neq i} |m_{ij}|\cdot|x_i x_j| \\
&\leq \sum_{i=1}^{N} m_{ii}\cdot x_i^2 + \sum_{i=1}^{N}\sum_{j\neq i} \sqrt{m_{ii}m_{jj}}\cdot|x_i x_j| \\
&\leq \sum_{i=1}^{N} m_{ii}\cdot x_i^2 + \sum_{i=1}^{N}\sum_{j\neq i} \frac{1}{2}(m_{ii}\cdot x_i^2 + m_{jj}\cdot x_j^2) \\
&= N\cdot\sum_{i=1}^{N} m_{ii}\cdot x_i^2 \\
&= N\cdot\|\mathbf{x}\|_{\mathcal{D}_M}.
\end{aligned} \tag{6.62}$$

Note that the second inequality is due to the positive definiteness of M, i.e., $|m_{ij}| \leq \sqrt{m_{ii}m_{jj}}$. Therefore, the proof is completed. □

- **Proof of Proposition** 6.2.1

By taking advantages of the defined norm $\|\cdot\|_{\mathcal{D}_M,\infty}$, the inequality (6.53) in Proposition 6.2.1 is equivalent to state that, with probability at least $1-\delta$,

$$\left\|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k\right\|_{\mathcal{D}_{\Sigma_k}^{-1/2},\infty} \leq \beta_k(\delta). \tag{6.63}$$

Therefore, we next prove the inequality (6.63) where $\beta_k(\delta)$ is defined in (6.54). Note that the Kalman consensus filter generates the state estimate $\widehat{\boldsymbol{\phi}}_k$ and covariance Σ_k as shown in (6.49), we first show, by the following lemma, an equivalent form of the Kalman consensus filter.

Lemma 6.2.2. *Suppose that $\widehat{\boldsymbol{\phi}}_k$ and covariance Σ_k are generated by (6.49), then at each iteration k, it is equivalent to write*

$$\Sigma_k = A[k:1]\Upsilon_k^{-1}A[k:1]^\top; \tag{6.64a}$$

$$\widehat{\boldsymbol{\phi}}_k = A[k:1]\Upsilon_k^{-1}\left(\Sigma_0^{-1}\widehat{\boldsymbol{\phi}}_0 + \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{z}_t\right), \tag{6.64b}$$

where the matrix $\Upsilon_k \in \mathbb{R}^{N\times N}$ *is defined as*

$$\Upsilon_k = \Sigma_0^{-1} + \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1} H_t A[t:1]. \tag{6.65}$$

Proof. Let us prove the above lemma by mathematical induction. First, it is straightforward to confirm that the above (6.64) is identical to the original recursion (6.49) when $k = 1$. Then, let us assume that (6.64) produces the same results as (6.49) up to the time-step k. Next, we prove the consistency for the time-step $k+1$.

Before proceeding, let us first notice the following identity with the definition of Υ_k,

$$\Upsilon_{k+1}^{-1} = (\mathbf{I}_N - \Upsilon_{k+1}^{-1} A[k:1]^\top H_k^\top V^{-1} H_k A[k:1])\Upsilon_k^{-1}. \tag{6.66}$$

Note that the above equality can be easily verified by multiplying Υ_{k+1} on the both sides.

Based on the recursion (6.49a), we plug in the previously obtained Σ_k in the form of (6.64a) and have that

$$\begin{aligned}
\Sigma_{k+1} &= A_{k+1}\Big(\Sigma_k^{-1} + H_k^\top V^{-1} H_k\Big)^{-1} A_{k+1}^\top \\
&= A_{k+1}\Big(A[k:1]^{-\top}\Upsilon_k A[k:1]^{-1} + H_k^\top V^{-1} H_k\Big)^{-1} A_{k+1}^\top \\
&= A[k+1:1]\Big(\Upsilon_k + A[k:1]^\top H_k^\top V^{-1} H_k A[k:1]\Big)^{-1} A[k+1:1]^\top \\
&= A[k+1:1]\Upsilon_{k+1}^{-1} A[k+1:1]^\top.
\end{aligned} \tag{6.67}$$

Similarly, we plug $\widehat{\boldsymbol{\phi}}_k$ in the form of (6.64b) into the recursion (6.49b) and obtain

$$\begin{aligned}
\widehat{\boldsymbol{\phi}}_{k+1} &= A_{k+1}\Big(\widehat{\boldsymbol{\phi}}_k + (\Sigma_k^{-1} + Y_k)^{-1}(\mathbf{y}_k - Y_k\widehat{\boldsymbol{\phi}}_k)\Big) \\
&= A_{k+1}\Big(\mathbf{I}_N - (\Sigma_k^{-1} + Y_k)^{-1} Y_k\Big)\widehat{\boldsymbol{\phi}}_k + A_{k+1}(\Sigma_k^{-1} + Y_k)^{-1} H_k^\top V^{-1}\mathbf{z}_k \\
&= A[k+1:1]\Big(\mathbf{I}_N - \Upsilon_{k+1}^{-1} A[k:1]^\top H_k^\top V^{-1} H_k A[k:1]\Big)\Upsilon_k^{-1} \\
&\quad \cdot \Big(\Sigma_0^{-1}\widehat{\boldsymbol{\phi}}_0 + \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{z}_t\Big) + A[k+1:1]\Upsilon_{k+1}^{-1} A[k:1]^\top H_k^\top V^{-1}\mathbf{z}_k \\
&= A[k+1:1]\Upsilon_{k+1}^{-1}\Big(\Sigma_0^{-1}\widehat{\boldsymbol{\phi}}_0 + \sum_{t=0}^{k} A[t:1]^\top H_t^\top V^{-1}\mathbf{z}_t\Big).
\end{aligned} \tag{6.68}$$

Note that the above identity (6.66) is applied in the last equality. Based on (6.67) and (6.68), the proof is completed.

□

Next, given that the state dynamics has $\boldsymbol{\phi}_k = A[k:1]\boldsymbol{\phi}_0$ and thus $\mathbf{z}_k = H_k A[k:1]\boldsymbol{\phi}_0 + \mathbf{n}_k$, the state estimate $\widehat{\boldsymbol{\phi}}_k$ can be further expressed as

$$\begin{aligned}\widehat{\boldsymbol{\phi}}_k &= A[k:1]\Upsilon_k^{-1}\Big(\Sigma_0^{-1}\widehat{\boldsymbol{\phi}}_0 + \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t + \Upsilon_k\boldsymbol{\phi}_0 - \Sigma_0^{-1}\boldsymbol{\phi}_0\Big) \\ &= \boldsymbol{\phi}_k + A[k:1]\Upsilon_k^{-1}\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t + A[k:1]\Upsilon_k^{-1}\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0).\end{aligned} \tag{6.69}$$

Therefore, it holds that $\forall \mathbf{x} \in \mathbb{R}^N$,

$$\begin{aligned}\mathbf{x}^\top(\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k) &= \mathbf{x}^\top A[k:1]\Upsilon_k^{-1}\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t + \mathbf{x}^\top A[k:1]\Upsilon_k^{-1}\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0) \\ &\overset{(6.1.a)}{\leq} \big\|A[k:1]^\top\mathbf{x}\big\|_{\Upsilon_k^{-1}} \cdot \Big\|\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t\Big\|_{\Upsilon_k^{-1}} + \big\|A[k:1]^\top\mathbf{x}\big\|_{\Upsilon_k^{-1}} \cdot \big\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\big\|_{\Upsilon_k^{-1}} \\ &\overset{(6.1.b)}{=} \big\|\mathbf{x}\big\|_{\Sigma_k} \cdot \Big(\Big\|\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t\Big\|_{\Upsilon_k^{-1}} + \big\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\big\|_{\Upsilon_k^{-1}}\Big) \\ &\overset{(6.1.c)}{\leq} N \cdot \big\|\mathbf{x}\big\|_{\mathcal{D}_{\Sigma_k}} \cdot \Big(\Big\|\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t\Big\|_{\Upsilon_k^{-1}} + \big\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\big\|_{\Upsilon_k^{-1}}\Big).\end{aligned} \tag{6.70}$$

where (6.1.a) is due to the Cauchy-Schwartz inequality; (6.1.b) is due to (6.64a); and (6.1.c) is based on Lemma 6.2.1.

Now, let $\mathbf{x} = \mathcal{D}_{\Sigma_k}^{-1}(\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k)$, it follows that

$$\big\|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k\big\|_{\mathcal{D}_{\Sigma_k}^{-1}} \leq N \cdot \Big(\Big\|\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t\Big\|_{\Upsilon_k^{-1}} + \big\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\big\|_{\Upsilon_k^{-1}}\Big). \tag{6.71}$$

According to the inequality in (6.60), we can have that

$$\begin{aligned}\big\|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k\big\|_{\mathcal{D}_{\Sigma_k}^{-1/2},\infty} &\leq \sqrt{N} \cdot \big\|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k\big\|_{\mathcal{D}_{\Sigma_k}^{-1}} \\ &\leq N^{3/2} \cdot \Big(\Big\|\sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t\Big\|_{\Upsilon_k^{-1}} + \big\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\big\|_{\Upsilon_k^{-1}}\Big).\end{aligned} \tag{6.72}$$

In order to prove the inequality (6.63), we now need to upper bound the two terms on the right hand side of (6.72); see the following two lemmas.

Lemma 6.2.3. *Let the conditions in Proposition 6.2.1 hold and the matrix Υ_k be defined as (6.65), then there exists a constant $c_1 = \|\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0\|/\sqrt{\underline{\sigma}}$ such that for $\forall k > 0$,*

$$\left\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\right\|_{\Upsilon_k^{-1}} \leq c_1. \tag{6.73}$$

Proof. By the definition (6.65) of the matrix Υ_k, it is straightforward to see that $\Upsilon_k^{-1} \leq \Sigma_0$, and therefore,

$$\begin{aligned}\left\|\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)\right\|_{\Upsilon_k^{-1}}^2 &= (\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)^\top \Sigma_0^{-1}\Upsilon_k^{-1}\Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0) \\ &\leq (\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0)^\top \Sigma_0^{-1}(\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0) \\ &\leq 1/\underline{\sigma} \cdot \|\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0\|^2,\end{aligned} \tag{6.74}$$

where the last inequality is due to the condition $\Sigma_0 \geq \underline{\sigma}\cdot\mathbf{I}_N$. Thus, the proof is completed. □

Lemma 6.2.4. *Let the conditions in Proposition 6.2.1 hold and the matrix Υ_k be defined as (6.65), then there exists a constant $c_2' = \bar{v}^2\sqrt{2N \cdot \max\{1, 1/\underline{v}\}}$ such that with probability at least $1-\delta$, for $\forall k > 0$,*

$$\left\| \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1}\mathbf{n}_t \right\|_{\Upsilon_k^{-1}} \leq c_2' \cdot \sqrt{\log\left(\frac{\bar{\sigma}/\underline{\sigma} + \bar{\alpha}\bar{\sigma}\cdot k/\underline{v}^2}{\delta^{2/N}}\right)}. \tag{6.75}$$

Proof. This proof is primarily based on the results presented in [163] (see Lemmas 8 – 10 and Theorem 1). For the notational simplicity, let us define

$$X_t := A[t:1]^\top H_t^\top V^{-1} \in \mathbb{R}^{N\times M}. \tag{6.76}$$

Then, according to Theorem 1 in [163], it holds with probability at least $1-\delta$ that,

$$\left\| \sum_{t=0}^{k-1} X_t\mathbf{n}_t \right\|_{\Omega_k^{-1}} \leq 2\bar{v}^2 \cdot \sqrt{\log\left(\frac{\det(\Omega_k)^{1/2}\det(\Sigma_0)^{1/2}}{\delta}\right)}, \tag{6.77}$$

where $\Omega_k := \Sigma_0^{-1} + \sum_{t=0}^{k-1} X_t X_t^\top \in \mathbb{R}^{N\times N}$. Let us recall the definition (6.65) of the matrix Υ_k and notice that there is a slight difference between Ω_k and Υ_k. Next, we show that there exists a constant $c_3' = \max\{1, 1/\underline{v}\}$ such that $\Omega_k \leq c_3' \cdot \Upsilon_k, \forall k > 0$. In fact, it holds that

$$\begin{aligned}\Omega_k &= \Sigma_0^{-1} + \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-2} H_t A[t:1] \\ &\leq \Sigma_0^{-1} + 1/\underline{v} \cdot \sum_{t=0}^{k-1} A[t:1]^\top H_t^\top V^{-1} H_t A[t:1] \\ &\leq \max\{1, 1/\underline{v}\} \cdot \Upsilon_k.\end{aligned} \tag{6.78}$$

Note that the first inequality is due to the fact that $\underline{v}$ is the smallest entry of the diagonal matrix V; see Assumption 6.2.2. Therefore, the previous statement can be immediately proved by letting $c_3' = \max\{1, 1/\underline{v}\}$, and based on this statement, it holds that $\Upsilon_k^{-1} \leq c_3' \cdot \Omega_k^{-1}$. Together with the inequality (6.77), one can have that

$$\Big\| \sum_{t=0}^{k-1} X_t \mathbf{n}_t \Big\|_{\Upsilon_k^{-1}} \leq \sqrt{c_3'} \cdot \Big\| \sum_{t=0}^{k-1} X_t \mathbf{n}_t \Big\|_{\Omega_k^{-1}} \leq 2\bar{v}^2 \sqrt{\max\{1, 1/\underline{v}\}} \cdot \sqrt{\log\Big(\frac{\det(\Omega_k)^{1/2} \det(\Sigma_0)^{1/2}}{\delta} \Big)}. \tag{6.79}$$

Moreover, according to the inequality of arithmetic and geometric means and the definition of Ω_k, it holds that

$$\det(\Omega_k) \leq \Big(1/N \cdot \mathrm{trace}\big(\Sigma_0^{-1}\big) + 1/N \cdot \sum_{t=0}^{k-1} \mathrm{trace}(X_t X_t^\top) \Big)^N, \tag{6.80}$$

where the trace of the matrix $X_t X_t^\top$ further has

$$\begin{aligned}\mathrm{trace}(X_t X_t^\top) &= \mathrm{trace}\Big(A[t:1]^\top H_t^\top V^{-2} H_t A[t:1] \Big) \\ &\overset{(6.2.a)}{\leq} 1/\underline{v}^2 \cdot \sum_{n=1}^{N} \mathbf{e}_n^\top A[t:1]^\top H_t^\top H_t A[t:1] \mathbf{e}_n \\ &\overset{(6.2.b)}{\leq} 1/\underline{v}^2 \cdot \sum_{n=1}^{N} \mathbf{e}_n^\top A[t:1]^\top A[t:1] \mathbf{e}_n \\ &\overset{(6.2.c)}{\leq} N \cdot \bar{\alpha}/\underline{v}^2.\end{aligned} \tag{6.81}$$

Note that (6.2.a) is due to $\underline{v} = \min_{i\in\mathcal{I}} v^i$ and $\mathbf{e}_n \in \mathbb{R}^N$ denotes the unit vector; (6.2.b) follows from the special form of the measurement matrix H_t, i.e., each row has only one element equal to one and all others equal to zero; and (6.2.c) is based on Assumption 6.2.1. In addition, given that the initialization Σ_0 ensures $\underline{\sigma}\cdot\mathbf{I}_N \leq \Sigma_0 \leq \bar{\sigma}\cdot\mathbf{I}_N$, it follows that $\mathrm{trace}(\Sigma_0^{-1}) \leq N/\underline{\sigma}$ and $\det(\Sigma_0) \leq \bar{\sigma}^N$. As a result, we can eventually arrive at

$$\begin{aligned}
&\sqrt{\log\Big(\det(\Omega_k)^{1/2}\det(\Sigma_0)^{1/2}/\delta\Big)} \\
&= \sqrt{1/2\cdot\log\Big(\det(\Omega_k)\Big) + 1/2\cdot\log\Big(\det(\Sigma_0)\Big) - \log(\delta)} \\
&\leq \sqrt{N/2}\cdot\sqrt{\log\Big(\frac{\bar{\sigma}/\underline{\sigma} + \bar{\alpha}\bar{\sigma}\cdot k/\underline{v}^2}{\delta^{2/N}}\Big)}.
\end{aligned} \tag{6.82}$$

Together with the inequality (6.77), the proof of Lemma 6.2.4 is completed.

□

Now, based on Lemmas 6.2.3 – 6.2.4 and inequality (6.72), it has been shown that, with probability $1-\delta$

$$\big\|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}_k\big\|_{\mathcal{D}_{\Sigma_k}^{-1/2},\infty} \leq N^{3/2}\cdot\left(c_1 + c_2'\cdot\sqrt{\log\Big(\frac{\bar{\sigma}/\underline{\sigma} + \bar{\alpha}\bar{\sigma}\cdot k/\underline{v}^2}{\delta^{2/N}}\Big)}\right), \tag{6.83}$$

with $c_1 = \|\widehat{\boldsymbol{\phi}}_0 - \boldsymbol{\phi}_0\|/\sqrt{\underline{\sigma}}$ and $c_2' = \bar{v}^2\sqrt{2N\cdot\max\{1, 1/\underline{v}\}}$. Therefore, Proposition 6.2.1 is proved.

- **Proof of Theorem** 6.2.3

Let us start the proof by introducing additional notations. Recall that $\mathbf{p}_k^\star$, as defined in (6.40), denotes the positions of the moving sources at time-step k, and similarly, $\mathbf{p}_k$ denotes the target positions for the multiple UAVs generated by our algorithm. To better characterize the positional information, let us define a mapping $\mathbf{a}(\cdot): \mathcal{S}^I \to \mathbb{R}^N$ which maps the position $\mathbf{p}$ to the N-dimensional vector,

$$\mathbf{a}(\mathbf{p}) = \sum_{i=1}^{I}\mathbf{e}_{s_i}, \tag{6.84}$$

where each s_i corresponds to the index of the position $\mathbf{p}[i]$. More precisely, since the positions $\mathbf{p}_k$ and $\mathbf{p}_k^\star$ are solved by the maximization problems; see (6.55) and (6.40), it can be immediately verified that the vectors $\mathbf{a}(\mathbf{p}_k)$ and $\mathbf{a}(\mathbf{p}_k^\star)$ must have I elements equal to one and all others equal to zero. Therefore, we denote the set of all possibilities of these vectors as

$$\mathcal{A} := \{\mathbf{a} \,|\, \mathbf{a} \in \{0,1\}^N, \mathbf{1}_N^\top \mathbf{a} = I\}. \tag{6.85}$$

Furthermore, for the notational simplicity, we abbreviate the above $\mathbf{a}(\mathbf{p}_k)$ and $\mathbf{a}(\mathbf{p}_k^\star)$ to $\mathbf{a}_k \in \mathcal{A}$ and $\mathbf{a}_k^\star \in \mathcal{A}$, respectively. With the help of these notations, the loss of function values can be expressed as,

$$r_k := F_k(\mathbf{p}_k^\star) - F_k(\mathbf{p}_k) = \langle \mathbf{a}_k^\star - \mathbf{a}_k, \boldsymbol{\phi}_k \rangle. \tag{6.86}$$

Next, we show, by the following lemma, that there exists an uniform upper bound for the loss of function values.

Lemma 6.2.5. *Suppose that Assumption 6.2.1 holds and the loss of function r_k is defined as (6.86), then there is an upper bound $\bar{\gamma} = 2\sqrt{I\bar{\alpha}} \cdot \|\boldsymbol{\phi}_0\|^2$ such that for $r_k \leq \bar{\gamma}$, $\forall k > 0$.*

Proof. Recall that the linear dynamics of the state $\boldsymbol{\phi}_k$ ensures $\boldsymbol{\phi}_k = A[k:1]\boldsymbol{\phi}_0$, thus based on (6.86), it follows that

$$\begin{aligned} r_k &\overset{(6.3.a)}{\leq} \|\mathbf{a}_k^\star - \mathbf{a}_k\| \cdot \|\boldsymbol{\phi}_k\| \\ &\overset{(6.3.b)}{\leq} \left(\|\mathbf{a}_k^\star\| + \|\mathbf{a}_k\|\right) \cdot \left\|\boldsymbol{\phi}_0^\top A[k:1]^\top A[k:1]\boldsymbol{\phi}_0\right\| \\ &\overset{(6.3.c)}{\leq} 2\sqrt{I\bar{\alpha}} \cdot \|\boldsymbol{\phi}_0\|^2 \end{aligned} \tag{6.87}$$

where (6.3.a) is due to the Cauchy-Schwartz inequality; (6.3.b) follows from the triangle inequality and the state dynamics; and (6.3.c) is based on the fact that both $\mathbf{a}_k^\star$ and $\mathbf{a}_k$ are from the set $\mathcal{A}$ as well as Assumption 6.2.1. □

Let us define another set $\boldsymbol{\chi}_k \in \mathbb{R}^N$ which is characterized by Proposition 6.2.1,

$$\boldsymbol{\chi}_k := \Big\{ \boldsymbol{\phi} \,\Big|\, \|\widehat{\boldsymbol{\phi}}_k - \boldsymbol{\phi}\|_{\mathcal{D}_{\Sigma_k}^{-1/2},\infty} \leq \beta_k(\delta) \Big\}. \tag{6.88}$$

It is guaranteed by Proposition 6.2.1 that $\boldsymbol{\phi}_k \in \boldsymbol{\chi}_k$ with probability at least $1-\delta$ at each iteration k.

With the help of the defined set $\boldsymbol{\chi}_k$, we now present a supporting lemma which measures the update of the target positions $\mathbf{p}_k$ (or $\mathbf{a}_k$) at each time-step k.

Lemma 6.2.6. *Under the conditions in Proposition 6.2.1, suppose that the positional information $\mathbf{a}_k$ is generated by solving the maximization problem (6.55) with the D-UCB $\boldsymbol{\mu}_k$ computed by (6.52), then the optimal function value $\langle \mathbf{a}_k, \boldsymbol{\mu}_k \rangle$ of (6.55) can be obtained by solving the following constrained bi-linear program,*

$$\max_{\mathbf{a}\in\mathcal{A},\, \boldsymbol{\phi}\in\boldsymbol{\chi}_k} \quad \langle \mathbf{a},\ \boldsymbol{\phi} \rangle. \tag{6.89}$$

In addition, it holds with probability $1-\delta$ that,

$$\langle \mathbf{a}_k,\ \boldsymbol{\mu}_k \rangle \geq \langle \mathbf{a}_k^\star,\ \boldsymbol{\phi}_k \rangle. \tag{6.90}$$

Proof. Notice that the program (6.89) can be written as the following equivalent form,

$$\max_{\mathbf{a}\in\mathcal{A}} \quad Q(\mathbf{a}), \tag{6.91}$$

where the objective function $Q(\cdot) : \mathcal{A} \to \mathbb{R}$ is defined by another maximization problem,

$$Q(\mathbf{a}) := \max_{\boldsymbol{\phi}\in\boldsymbol{\chi}_k} \quad \langle \mathbf{a},\ \boldsymbol{\phi} \rangle. \tag{6.92}$$

Based on the KKT conditions and the definition of the feasible set $\boldsymbol{\chi}_k$, the optimal solution $\boldsymbol{\phi}^\star$ of the problem (6.92) can be analytically expressed as

$$\boldsymbol{\phi}^\star = \widehat{\boldsymbol{\phi}}_k + \beta_k(\delta) \cdot \mathrm{diag}^{1/2}(\Sigma_k), \tag{6.93}$$

which is exactly the same as the definition of D-UCB in (6.52). Therefore, it holds that

$$\langle \mathbf{a}_k,\ \boldsymbol{\mu}_k \rangle = \max_{\mathbf{a}\in\mathcal{A},\, \boldsymbol{\phi}\in\boldsymbol{\chi}_k} \langle \mathbf{a},\ \boldsymbol{\phi} \rangle. \tag{6.94}$$

Furthermore, since Proposition 6.2.1 guarantees that $\boldsymbol{\phi}_k \in \boldsymbol{\chi}_k$ with probability $1-\delta$ and $\mathbf{a}^\star = \arg\max_{\mathbf{a}\in\mathcal{A}} \langle \mathbf{a},\ \boldsymbol{\phi}_k \rangle$, it is easy to verify that (6.90) holds with probability $1-\delta$. □

Now, we are ready to prove the statement in Theorem 6.2.3, i.e., $\sum_{k=1}^{K} r_k \leq \mathcal{O}(\sqrt{K}\log K)$. Before proceeding, let us first recall that the vector norm $\|\cdot\|_{\mathcal{D}_M,1}$ as defined in (6.59) is the dual norm of $\|\cdot\|_{\mathcal{D}_M^{-1},\infty}$ as defined in (6.58). Therefore, the loss of function value r_k has

$$\begin{aligned}
r_k &= \langle \mathbf{a}_k^\star,\ \boldsymbol{\phi}_k \rangle - \langle \mathbf{a}_k,\ \boldsymbol{\phi}_k \rangle \\
&\overset{(46..a)}{\leq} \langle \mathbf{a}_k,\ \boldsymbol{\mu}_k \rangle - \langle \mathbf{a}_k,\ \boldsymbol{\phi}_k \rangle \\
&\overset{(6.4.b)}{\leq} \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}^{1/2},1} \cdot \|\boldsymbol{\mu}_k - \boldsymbol{\phi}_k\|_{\mathcal{D}_{\Sigma_k}^{-1/2},\infty} \\
&\overset{(6.4.c)}{\leq} 2\beta_k(\delta) \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}^{1/2},1} \\
&\overset{(6.4.d)}{\leq} 2\sqrt{N}\beta_k(\delta) \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}},
\end{aligned} \tag{6.95}$$

where the inequality (6.4.a) is due to the above Lemma 6.2.6; (6.4.b) follows from the Hölder's inequality; (6.4.c) is due to the triangle inequality and the fact that both $\boldsymbol{\mu}_k$ and $\boldsymbol{\phi}_k$ are in the set $\boldsymbol{\chi}_k$; and (6.4.d) comes from the inequality (6.60). Next, to further investigate the key term $\|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_{k-1}}}$, we show an upper bound for the cumulative $\|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_{k-1}}}$'s.

Lemma 6.2.7. *Suppose that the conditions in Proposition 6.2.1 hold and the positional information $\mathbf{a}_k$'s are generated by Algorithm 10, then it holds that for $\forall K > 0$,*

$$\sum_{k=0}^{K-1} \min\{1, 1/\bar{v}\cdot\|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}}^2\} \leq 2N \cdot \log\Big(\det(\Sigma_0)^{1/N} \cdot \bar{\alpha} \cdot \big((\underline{\alpha}\underline{\sigma})^{-1} + K\cdot(\underline{\alpha}\underline{v})^{-1}\big)\Big). \tag{6.96}$$

Proof. Recall that the matrix Σ_k is generated by the following recursion,

$$\Sigma_{k+1} = A_{k+1}\Big(\Sigma_k^{-1} + H_k^\top V^{-1} H_k\Big)^{-1} A_{k+1}^\top. \tag{6.97}$$

For the sake of presentation, let us first focus on the inverse of Σ_k, i.e., $\Theta_k = \Sigma_k^{-1} \in \mathbb{R}^{N\times N}$, and thus it holds that,

$$\Theta_{k+1} = A_{k+1}^{-\top}\Big(\Theta_k + H_k^\top V^{-1} H_k\Big) A_{k+1}^{-1}. \tag{6.98}$$

Consider the determinant of the matrices Θ_k's, one can have that

$$\begin{aligned}
\det(\Theta_{k+1}) &= 1/\det(A_{k+1}^\top A_{k+1}) \cdot \det\Big(\Theta_k + H_k^\top V^{-1} H_k\Big) \\
&= 1/\det(A_{k+1}^\top A_{k+1}) \cdot \det\Big(\Theta_k^{1/2}\Big(\mathbf{I}_N + \Theta_k^{-1/2} H_k^\top V^{-1} H_k \Theta_k^{-1/2}\Big)\Theta_k^{1/2}\Big) \\
&= \det(\Theta_k)/\det(A_{k+1}^\top A_{k+1}) \det\Big(\mathbf{I}_N + \Theta_k^{-1/2} H_k^\top V^{-1} H_k \Theta_k^{-1/2}\Big).
\end{aligned} \tag{6.99}$$

For simplicity, we here use Y_k to substitute $H_k^\top V^{-1} H_k$ again. Consider that the noise covariance matrix V is diagonal and H_k takes the specific form of $H_k = [\mathbf{e}_l]_{l\in\cup_{i=1}^I \mathcal{C}^i}^\top$, where each set $\mathcal{C}^i$ contains the indices of the positions covered by the i-th UAV's sensing area. Therefore, the matrix Y_k is also diagonal and can be expressed as

$$Y_k = \sum_{i=1}^{I} \sum_{l\in\mathcal{C}^i} 1/v^i \cdot \mathbf{e}_l \mathbf{e}_l^\top. \tag{6.100}$$

Further, let us denote $\Theta_k^{-1/2} Y_k \Theta_k^{-1/2}$ by $\Xi_k \in \mathbb{R}^{N\times N}$. Suppose that $\lambda_n(\Xi_k)$ represents the n-th eigenvalue and ξ_{nn}^k is the n-th diagonal entry of Ξ_k, then the trace of the matrix has

$$\mathrm{trace}(\Xi_k) = \sum_{n=1}^{N} \lambda_n(\Xi_k) = \sum_{n=1}^{N} \xi_{nn}^k. \tag{6.101}$$

In addition, we denote $\boldsymbol{\theta}_n^k \in \mathbb{R}^N$ the n-th column of the matrix $\Theta_k^{-1/2}$; note that $(\boldsymbol{\theta}_n^k)^\top$ is also the n-th row since $\Theta_k^{-1/2}$ is symmetric. As a result of the specific structure of the matrix Y_k, the diagonal entries ξ_{nn}^k of Ξ_k has

$$\begin{aligned}
\xi_{nn}^k &\overset{(6.5.a)}{=} \Big(\sum_{i=1}^{I} \delta_k^i(n)/v^i\Big) \cdot (\boldsymbol{\theta}_n^k)^\top \boldsymbol{\theta}_n^k \\
&\overset{(6.5.b)}{=} \Big(\sum_{i=1}^{I} \delta_k^i(n)/v^i\Big) \sigma_{nn}^k \overset{(6.5.c)}{\geq} 1/\bar{v} \cdot \sum_{i=1}^{I} \delta_k^i(n) \sigma_{nn}^k,
\end{aligned} \tag{6.102}$$

where in (6.5.a), we let $\delta_k^i(n) = 1$ if the position indexed by n is in the sensing area $\mathcal{C}^i$ at the time-step k, and $\delta_n^i = 0$ otherwise; (6.5.b) is due to the definition of $\boldsymbol{\theta}_n^k$ and the fact that σ_{nn}^k denotes the n-th diagonal entry of Σ_k; and (6.5.c) comes from the fact that $\bar{v} = \max_{i\in\mathcal{I}} v^i$. Now, based on (6.102), one can further have

$$\begin{aligned}\sum_{n=1}^{N} \xi_{nn}^k &\geq 1/\bar{v} \cdot \sum_{n=1}^{N}\sum_{i=1}^{I} \delta_k^i(n)\sigma_{nn}^k \\ &\overset{(6.6.a)}{\geq} 1/\bar{v} \cdot \sum_{i=1}^{I} \mathbf{e}_{s_k^i}^\top \Sigma_k \mathbf{e}_{s_k^i} \\ &\overset{(6.6.b)}{=} 1/\bar{v} \cdot \mathbf{a}_k^\top \mathcal{D}_{\Sigma_k} \mathbf{a}_k \overset{(6.6.c)}{=} 1/\bar{v} \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}}^2,\end{aligned} \tag{6.103}$$

where s_k^i denotes the index of the i-th UAV's position at the time-step k in (6.6.a) and $\delta_k^i(s_k^i)$ must be one; (6.6.b) is by the definition (6.84) of $\mathbf{a}_k$ and (6.6.c) is by the definition of $\|\cdot\|_{\mathcal{D}_{\Sigma_k}}$.

Now, the previous equalities in (6.99) can be continued as

$$\begin{aligned}\det(\Theta_{k+1}) &= \det(\Theta_k)/\det(A_{k+1}^\top A_{k+1}) \cdot \det(\mathbf{I}_N + \Xi_k) \\ &\overset{(7.a)}{=} \det(\Theta_k)/\det(A_{k+1}^\top A_{k+1}) \cdot \prod_{n=1}^{N}\Big(1 + \lambda_n(\Xi_k)\Big) \\ &\overset{(7.b)}{\geq} \det(\Theta_k)/\det(A_{k+1}^\top A_{k+1}) \cdot \Big(1 + \sum_{n=1}^{N} \lambda_n(\Xi_k)\Big) \\ &\overset{(7.c)}{=} \det(\Theta_k)/\det(A_{k+1}^\top A_{k+1}) \cdot \Big(1 + \sum_{n=1}^{N} \xi_{nn}^k\Big) \\ &\overset{(7.d)}{\geq} \det(\Theta_k)/\det(A_{k+1}^\top A_{k+1}) \cdot \Big(1 + 1/\bar{v} \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}}^2\Big),\end{aligned} \tag{6.104}$$

where (7.a) is due to the fact that the determinant of a matrix equals the product of eigenvalues; (7.b) follows from the inequality of arithmetic and geometric means and the positive definiteness of the matrix Ξ_k; (7.c) is based on the equality (6.101); and (7.d) is due to the inequality (6.103). Subsequently, applying (6.104) recursively yields

$$\begin{aligned}\det(\Theta_{k+1}) &\geq \det(\Theta_0)/\det\Big(A[k+1:1]^\top A[k+1:1]\Big) \cdot \prod_{t=0}^{k}\Big(1 + 1/\bar{v} \cdot \|\mathbf{a}_t\|_{\mathcal{D}_{\Sigma^t}}^2\Big) \\ &\geq \bar{\alpha}^{-N} \det(\Theta_0) \cdot \prod_{t=0}^{k}\Big(1 + 1/\bar{v} \cdot \|\mathbf{a}_t\|_{\mathcal{D}_{\Sigma^t}}^2\Big).\end{aligned} \tag{6.105}$$

Note that the last inequality relies on Assumption 6.2.1.

Next, notice that $\min\{1, x\} \leq 2\log(1+x)$ is always true for any scalar $x \geq 0$, therefore,

$$\sum_{t=0}^{k} \min\{1, 1/\bar{v} \cdot \|\mathbf{a}_t\|^2_{\mathcal{D}_{\Sigma^t}}\} \leq \sum_{t=0}^{k} 2\log\left(1 + 1/\bar{v} \cdot \|\mathbf{a}_t\|^2_{\mathcal{D}_{\Sigma^t}}\right) \leq 2\log\left(\bar{\alpha}^N \cdot \det(\Theta_{k+1})/\det(\Theta_0)\right). \tag{6.106}$$

Furthermore, based on the recursion (6.98) of Θ_k, it follows that

$$\Theta_{k+1} = A[k+1:1]^{-\top}\Theta_0 A[k+1:1]^{-1} + \sum_{t=0}^{k} A[k+1:t+1]^{-\top} H_t V^{-1} H_t A[k+1:t+1]^{-1}, \tag{6.107}$$

Thus, one can have that

$$\begin{aligned}
\det(\Theta_{k+1}) &\leq \left(1/N \cdot \operatorname{trace}(\Theta_{k+1})\right)^N \\
&= \left(1/N \cdot \sum_{i=1}^{N} \mathbf{e}_n^\top \Theta_{k+1} \mathbf{e}_n\right)^N \\
&= \Bigg(1/N \cdot \sum_{i=1}^{N} \Big(\mathbf{e}_n^\top A[k+1:1]^{-\top}\Theta_0 A[k+1:1]^{-1}\mathbf{e}_n \\
&\quad + \sum_{t=0}^{k} \mathbf{e}_n^\top A[k+1:t+1]^{-\top} H_t V^{-1} H_t A[k+1:t+1]^{-1}\mathbf{e}_n\Big)\Bigg)^N \\
&\leq \left(1/N \cdot \sum_{i=1}^{N} \left((\underline{\alpha}\underline{\sigma})^{-1} + \sum_{t=0}^{k} (\underline{\alpha}\underline{v})^{-1}\right)\right)^N \\
&= \left((\underline{\alpha}\underline{\sigma})^{-1} + (k+1)\cdot(\underline{\alpha}\underline{v})^{-1}\right)^N.
\end{aligned} \tag{6.108}$$

Note that the last inequality comes from i) $\Sigma_0 \leq \underline{\sigma} \cdot \mathbf{I}_N$; ii) $A[k:t]^{-\top} A[k:t]^{-1} \leq \underline{\alpha}^{-1} \cdot \mathbf{I}_N$ (see Assumption 6.2.1); and iii) $H_t^\top V^{-1} H_t \leq \underline{v}^{-1} \cdot \mathbf{I}_N$ since the specific form of H_t and $\underline{v} = \min_{i\in\mathcal{I}} v^i$. As a consequence, it holds that

$$\log\left(\bar{\alpha}^N \cdot \det(\Theta_{k+1})/\det(\Theta_0)\right) \leq N \cdot \log\left(\det(\Sigma_0)^{1/N} \cdot \bar{\alpha}\Big((\underline{\alpha}\underline{\sigma})^{-1} + (k+1)\cdot(\underline{\alpha}\underline{v})^{-1}\Big)\right). \tag{6.109}$$

Together with the inequality (6.106), the proof is completed. □

With the help of the above Lemma 6.2.7, we can now continue our proof for the theorem. Since Lemma 6.2.5 has guaranteed that the loss of function $r_k \leq \bar{\gamma} = 2\sqrt{I\bar{\alpha}} \cdot \|\boldsymbol{\phi}_0\|^2, \forall k > 0$ Based on the inequality (6.95), it follows that

$$\begin{aligned} r_k &\leq \min\left\{\bar{\gamma},\ 2\sqrt{N}\beta_k(\delta) \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}}\right\} \\ &\leq \kappa \cdot \min\left\{1,\ 2\sqrt{N}\beta_k(\delta)/\sqrt{\bar{v}} \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}}\right\} \\ &\leq \kappa\beta_k(\delta) \cdot \min\left\{1,\ 1/\sqrt{\bar{v}} \cdot \|\mathbf{a}_k\|_{\mathcal{D}_{\Sigma_k}}\right\}. \end{aligned} \tag{6.110}$$

In the last two inequalities, we let $\kappa = \max\{\bar{\gamma},\ \sqrt{\bar{v}}\}$ and $\beta_k(\delta) = \max\{1,\ 2\sqrt{2}\beta_k(\delta)\}$. According to the definition (6.54) of the non-decreasing sequence $\{\beta_k(\delta)\}_{k\in\mathbb{N}_+}$, it can be confirmed that the sequence $\{\beta_k(\delta)\}_{k\in\mathbb{N}_+}$ is non-decreasing, i.e., $\beta_k(\delta) \leq \beta_{k+1}(\delta)$. Then, one can have

$$\begin{aligned} \sum_{k=0}^{K-1} r_k &\leq \sqrt{K \cdot \sum_{k=0}^{K-1} r_k^2} \\ &\overset{(8.a)}{\leq} \kappa\beta_K(\delta) \cdot \sqrt{K \cdot \sum_{k=0}^{K-1} \min\left\{1,\ 1/\bar{v} \cdot \|\mathbf{a}_k\|^2_{\mathcal{D}_{\Sigma_k}}\right\}} \\ &\overset{(8.b)}{\leq} \kappa\beta_K(\delta) \cdot \sqrt{2KN} \cdot \sqrt{\log\left(\det(\Sigma_0)^{1/N} \cdot \bar{\alpha}\Big((\underline{\alpha}\underline{\sigma})^{-1} + K \cdot (\underline{\alpha}\underline{v})^{-1}\Big)\right)}, \end{aligned} \tag{6.111}$$

where $(8.a)$ follows from the inequality (6.110) and $(8.b)$ is due to Lemma 6.2.7. Given that $\beta_K(\delta) = \max\{1,\ 2\sqrt{2}\beta_K(\delta)\}$ and $\beta_K(\delta) = \mathcal{O}(\sqrt{\log K})$ in Proposition 6.2.1, it can be obtained either $\beta_K(\delta) = 1$ or $\beta_K(\delta) = \mathcal{O}(\sqrt{\log K})$. Therefore, together with the inequality (6.111), the statement in Theorem 6.2.3 is proved, i.e., $\sum_{k=0}^{K} r_k \leq \mathcal{O}(\sqrt{K}\log K)$.

Remark 6.2.3. *A significant difference between the classical linear upper confidence bound (UCB) algorithm [164] and the present one is that we construct the D-UCB, rather than the standard UCB, to drive the update of* $\mathbf{p}_k$*'s. Due to this difference, one cannot immediately prove the above Theorem 6.2.3 by following exactly the steps in [164]. A remarkable idea of our proof is to define a specific vector norm which interplays with the form of D-UCB and then establish the regret analysis with respect to the specific norm. This makes our theoretical results non-trivial. In addition, we should also emphasize that the introduction of D-UCB helps reducing the computational complexity of our algorithm significantly, when solving the*

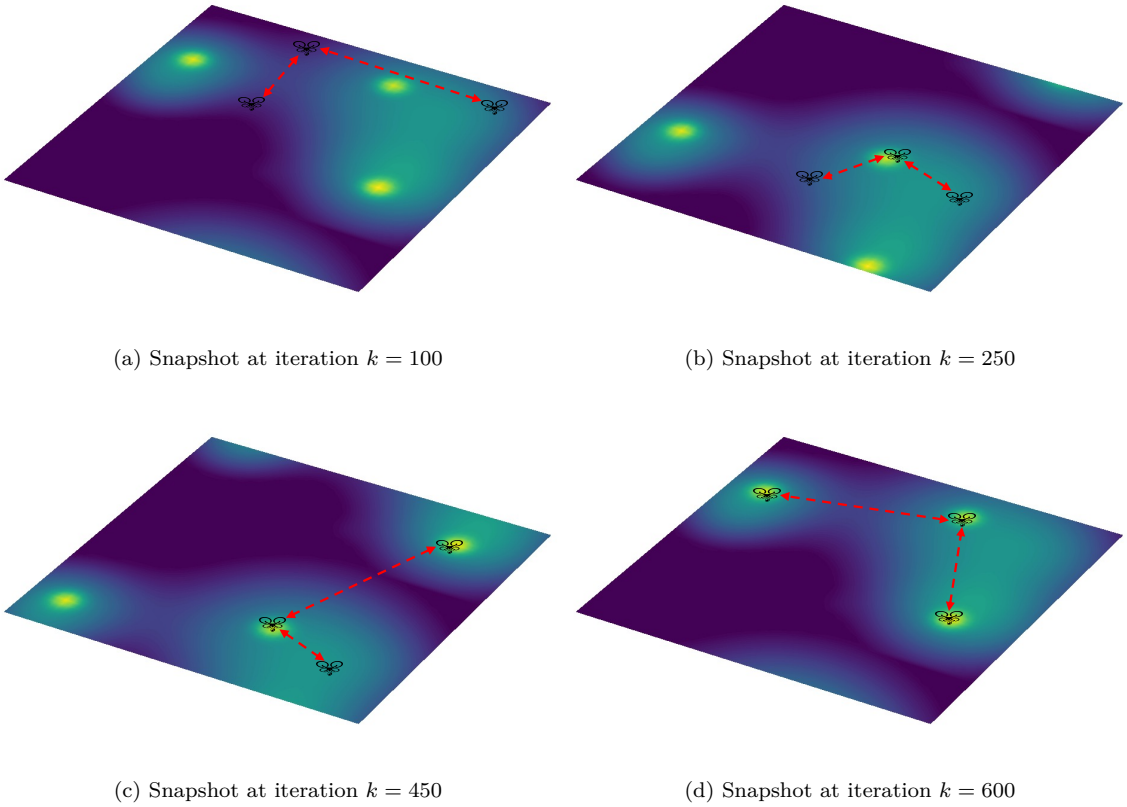
problem in the multi-UAV setting. Since the standard UCB is defined in a joint sense, when solving the multi-UAV maximization problem (6.55) with the standard UCB, it is inherently a combinatorial optimization and can be extremely complicated to find the exact solution. In contrast, due to the fact that the D-UCB takes the upper bounds marginally as mentioned before, the maximization (6.55) can be essentially decomposed and becomes much easier to solve for exact solutions. We remark this as one of the most important contributions of the proposed algorithm.

Remark 6.2.4. *It is stated in Theorem 6.2.3 that our algorithm can generate a sequence of the target positions $\mathbf{p}_k$'s such that the cumulative regret, i.e., the cumulative loss of function values $F_k(\mathbf{p}_k^\star) - F_k(\mathbf{p}_k)$, is upper bounded sub-linearly. Given the fact that the function $F_k(\cdot)$ is defined on a finite set $\mathcal{S}^I$; see definition in Section 6.2.1, together with the loss of function value must be positive, one can immediately conclude that the sequence of $\mathbf{p}_k$ converges to the optimal solution $\mathbf{p}_k^\star$. This also implies that the steady state is reached in which the multi-UAV system manages to tracking the top I moving sources.*

6.2.4 Numerical Experiment

We demonstrate the effectiveness of our distributed algorithm, by considering a real-world methane leaking source seeking problem using the multi-UAV system. In fact, such a problem has been broadly studied in the area of robotics; see e.g., [165], [166]. Compared to these existing works, a primary difference here is that we deploy multiple UAVs, rather than a single one, to the target methane field. As a result, we expect that the individual UAV will be able to track the distinct and possibly moving leaking sources by leveraging the cooperations among the entire team of UAVs.

Let us suppose that the target methane field is described by a $D \times D$ lattice, as shown in the background of Fig. 6.8. Each cell $l \in \{1, 2, \cdots, D^2\}$ in the lattice is represented by its position $\mathbf{s}^l$ and also the quantity $\phi_t(\mathbf{s}^l)$ which indicates the level of methane concentration at the time-step t. Overall, the N-dimensional vector $\boldsymbol{\phi}_t = [\phi_t(\mathbf{s}^1), \phi_t(\mathbf{s}^2), \cdots, \phi_t(\mathbf{s}^N)]^\top$ with $N = D^2$ characterizes the state of the entire methane field of interest. More specifically, in this simulation, we set the size of the methane field as $D = 50$. The initialized methane



(a) Snapshot at iteration $k = 100$

(b) Snapshot at iteration $k = 250$

(c) Snapshot at iteration $k = 450$

(d) Snapshot at iteration $k = 600$

Figure 6.8. Demonstration of three UAVs' tracking of the moving leaking sources in an unknown methane field

state $\boldsymbol{\phi}_0$ is generated through Gaussian kernels with leaking sources having largest concentrations among the field, and then we let leaking sources move within the field so that the time-varying $\boldsymbol{\phi}_t$ is generated. In order to explore the unknown target methane field and furthermore track the moving leaking sources, we employ a team of three UAVs, each of them equipped with a sensor that is capable of measuring a circular area with radius $r = 3$; see the detailed measurement model (6.44) and the description of measurement matrix (6.46) in Remark 6.2.2. In particular, we assume that the sensing noise of each UAV is independent and identically distributed Gaussians with zero-mean and covariance $V^i = \mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix with appropriate dimension. Note that, since the maximum value of the state $\boldsymbol{\phi}_t$ is set around 5, the noise covariance is reasonably large so that the overall problem is not trivial to solve. Besides, it is also assumed that the three UAVs can exchange information with their immediate neighbors, and the communication channels, shown as the red dot lines in Fig. 6.8, follow a simple undirected connected graph.

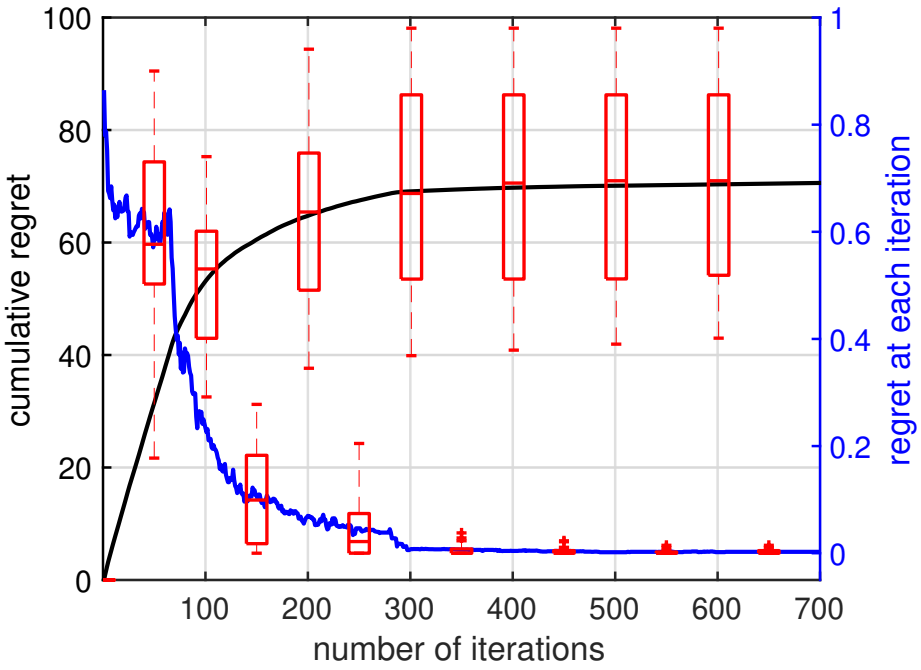
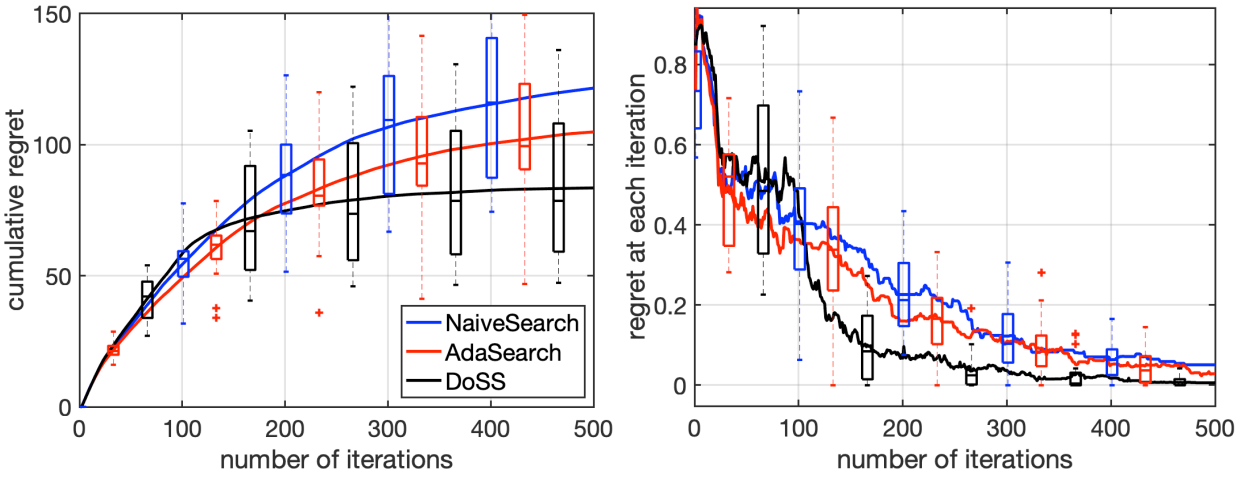


Figure 6.9. Regret analysis

To demonstrate the result of tracking of the moving leaking sources, four snapshots are taken and shown in Fig. 6.8 at the iterations $k = 100, 250, 450, 600$, respectively. It can be observed that the team of UAVs is able to locate all three moving leaking sources at the 600-th iteration. In addition, in order to show the simulation result quantitatively, Fig. 6.9 plots both the regret $r_k = F_k\big(\mathbf{p}^\star(k)\big) - F_k\big(\mathbf{p}_k\big)$ at each iteration as a blue line, and the cumulative regret $\sum_{t=1}^{k} r_t$ as a black line. Note that each line is obtained by the data averaged from 20 Monte-Carlo trials; the standard deviation is also reported in the figure. It can be concluded from Fig. 6.9 that the regret r_k decreases to zero as the number of iterations grows, which confirms that the team of UAVs will be able to track the moving leaking sources. Besides, the cumulative regret shows a sub-linear increase, which is also consistent with the theoretical result of Theorem 6.2.3.

In order to validate the effectiveness of the proposed algorithm, we further compare the performance of our algorithm, termed as `DoSS`, with two benchmark schemes: i) the `AdaSearch` algorithm [158]; and ii) a naive approach, termed as `NaiveSearch`, in which the UAVs scan the whole unknown field repeatedly and determine the position of leaking



(a) Cumulative regret

(b) Regret at each iteration

Figure 6.10. Comparison of the regret with three different schemes

sources by the current estimation of the field. Note that, to evaluate these three scheme fairly, we here only consider a static target methane field. As previously, we run each of the three schemes for 20 Monte-Carlo trials, and Fig. 6.10 shows the simulation results. It can be observed from this figure that our `DoSS` algorithm outperforms both `AdaSearch` and `NaiveSearch` algorithms in terms of the regret descent rate, which means that the our algorithm can locate the leaking sources more efficiently in an unknown methane field than the two others.

7. CONCLUSIONS AND FUTURE DIRECTIONS

7.1 Conclusions

Focusing on the two key aspects of the distributed processing, i.e., communication and computation, this dissertation develops a suite of new algorithms for solving the distributed deterministic/stochastic optimization problems, studies convergence properties of the proposed algorithms, and explores the opportunities of applying the idea of distributed optimization into real-world applications especially with multi-UAV systems.

- **Generalized Algorithmic Frameworks for Solving Distributed Deterministic Optimization.** Starting from the perspectives of communication and computation respectively, Chapters 2 and 3 propose two generalized algorithms, namely the DGDx algorithm and the NetProx algorithm. Motivated by improving the inexact convergence of the standard DGD iteration, the DGDx algorithm incorporates multiple communication rounds into each consensus step and ensures the sufficiently accurate result via a general inexact consensus operator. Instead of explicitly specifying the number of communication rounds according to the iteration index k, we directly control the consensus accuracy through the introduced consensus error bound. It is shown that the exact convergence can be achieved with a constant step-size when the sequence of error bounds decays to zero. In addition, the NetProx algorithm generalizes the choice of local approximation functions for each agent, observing that the standard DGD algorithm is just a special case when the linear approximation is adopted. More precisely, it is shown that, with the higher order approximation functions, the convergence of the NetProx algorithm will be accelerated, but meanwhile, it will cost more computational resources for each agent at each iteration. It can be concluded that both of the two algorithms offer the great potential to balance the communication and computation in distributed optimization.

- **Resilient Distributed Min-Max Optimization under Network Communication Attacks.** Chapter 4 investigates the resilience of distributed algorithms when communication attacks are potentially present within the multi-agent network. In par-

ticular, a special instance of the distributed optimization problem – distributed min-max optimization, is studied in this chapter. Building on the idea of resilient convex combination which helps to eliminate the malicious information injected by the unidentifiable communication attacks, a consensus-based distributed algorithm is proposed for solving the min-max problem. As distinct from the majority of existing works which often compromise the global objective to some extent while designing their resilient solution methods, the proposed algorithm takes into account all the agents within the network no matter whose communication channels are attacked or non-attacked. It is shown that, under some reasonable assumptions, e.g., the attacked communication channels can be recovered within a certain time-window, the proposed algorithm converges to the exact global optimal solution which is quite challenging or even impossible to achieve for the existing resilient distributed algorithms.

- **Adapting the PH Method under the Distributed Framework for Solving Two-Stage Stochastic Programs.** Chapter 5 further explores the potential of using distributed computing techniques to the solve notoriously hard stochastic optimization problems. A DistPH algorithm is developed to solve the two-stage stochastic programs under a peer-to-peer multi-agent network. Similar to existing parallel frameworks, since individual agents only need to take charge of the subproblems corresponding to a single or a few sampled scenarios, the computational burden of the distributed algorithm is spread over the entire network and thus reduced for each agent. However, unlike the parallel frameworks for which a master node is always required to provide central coordination for workers, the DistPH algorithm eliminates such requirement and only builds on peer communications over a more general connected network. Consequently, the proposed distributed algorithm is expected to be more advantageous against the parallel PH method, since the communication burden for the master node is also spread over the entire network. It is proved that exact convergence can be theoretically guaranteed when the considered stochastic problem has only continuous decision variables subject to convex constraints. Moreover, several computational issues are also investigated for the mixed-integer cases to further improve the algorithm efficiency.

- **Application I: Distributed Data Fusion for On-Scene Signal Sensing with Multi-UAV Systems.**

  Real-world applications with multi-UAV systems are also explored in Chapter 6 by applying the developed distributed optimization techniques. The first application concerns a health-care scenario where the health status of a certain target in rural environment needs to be monitored by sensing some vital on-scene biological signals, during or after an incident. Consider that, due to the complexity of biological signals and the limitation of UAVs such as stabilization requirements and payloads, the sensory data from each individual UAV might not be accurate enough to conduct a further diagnosis. Thus, a distributed data fusion framework is proposed to integrate the various measurements from multiple UAVs. As an important building block of the proposed data fusion algorithm, the average/sum consensus scheme developed in the previous chapters is adopted. Both theoretical analysis and numerical simulations show that our algorithm has great potential to be applied in the multi-UAV rural health monitoring applications.

- **Application II: Distributed Source Tracking in Unknown Environments with Multi-UAV Systems.**

  The second application considers a multi-UAV source tracking mission in an unknown and dynamical environment, in which a group of UAVs is deployed and expect to dynamically locate as many as local sources as possible. By applying the previous average/sum consensus scheme again, we develop an adaptive on-line framework which combines both estimation of the unknown environment and task planning for the multi-UAV system simultaneously. Additionally, a novel notion of dummy confidence upper bound is introduced to significantly reduce the computational complexity in solving the subproblems of multi-UAV task planning in the distributed manner. The performance of the proposed algorithm is theoretically guaranteed by showing a sub-linear upper bound of the cumulative regret. Numerical results on a real-world methane emission tracking problem also demonstrate the effectiveness of the proposed algorithm.

7.2 Future Directions

Future directions that can primarily extend the work in this dissertation are summarized as follows.

- **Design of Optimal Distributed Algorithms with Communication and Computation Budget.** The proposed DGDx and NetProx algorithms inherently generalize the standard DGD algorithm from the perspectives of communication and computation, respectively, and thus offer the flexibility to balance the two types of cost in solving specific optimization problems. For instance, with a distributed system whose communication resources are more affordable than computation, then the DGDx algorithm would be a more promising choice since multiple communication rounds can be incorporated into the implementation of algorithm. Conversely, the NetProx would be preferred since the higher order approximation function can be adopted to accelerate the convergence. Based on such an idea, a natural question to ask is how to design an optimal distributed algorithm by given the communication and computation budget? The major challenge here might be how to define the convergence rate in a broader notion which should be quantified by both two types of cost.

- **Resilient Distributed Algorithm for Solving General Optimization Problems.** Chapter 4 presents the resilient algorithm for solving the specific distributed min-max optimization problem. The main contribution of the proposed algorithm is its convergence to the exact optimal solution which takes into account all agents' local objective functions. Nevertheless, how to achieve such a goal in solving the general distributed optimization is still an open research problem. The existing works [18], [19] compromise the global objective to some extent while designing their approaches. One can first try to enhance the resilient convex combination scheme in Section 4.2 to guarantee a pure average consensus, and then many distributed algorithm can be applied to solve the general optimization problem.

- **Integrating Network Sampling Techniques with the Distributed PH Method.** Chapter 5 studies the two-stage stochastic program in which the uncertainty is readily

represented by a finite number of scenarios. When the uncertainty follows a continuous distribution, one may need to take the sampling of random variables into consideration, and the distributed optimization will be expanded into three dimensions, i.e. communication, computation, and sampling. Leveraging the idea of network sampling, one might be able to greatly enhance the proposed DistPH algorithm by further reducing the communication and computation workload. In this case, both first-order and second-order moment properties need to be investigated. It is expected to achieve the linear convergence rate in expectation and also the variance reduction for the randomized solution.

REFERENCES

[1] J. A. Bazerque and G. B. Giannakis, "Distributed spectrum sensing for cognitive radio networks by exploiting sparsity," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1847–1862, 2009.

[2] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links—Part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350–364, 2007.

[3] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, "Consensus in ad hoc WSNs with noisy links—Part II: Distributed estimation and smoothing of random signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, 2008.

[4] S. B. Al Islam and A. Hajbabaie, "Distributed coordinated signal timing optimization in connected transportation networks," *Transportation Research Part C: Emerging Technologies*, vol. 80, pp. 272–285, 2017.

[5] M. Wang, W. Daamen, S. P. Hoogendoorn, and B. van Arem, "Cooperative car-following control: Distributed algorithm and impact on moving jam features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1459–1471, 2015.

[6] B. Chen and H. H. Cheng, "A review of the applications of agent technology in traffic and transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 485–497, 2010.

[7] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 32–43, 2014.

[8] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[9] E. P. Xing, Q. Ho, W. Dai, J. K. Kim, J. Wei, S. Lee, X. Zheng, P. Xie, A. Kumar, and Y. Yu, "Petuum: A new platform for distributed machine learning on big data," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 49–67, 2015.

[10] G. B. Giannakis, V. Kekatos, N. Gatsis, S.-J. Kim, H. Zhu, and B. F. Wollenberg, "Monitoring and optimization for power grids: A signal processing perspective," *IEEE Signal Processing Magazine*, vol. 30, no. 5, pp. 107–128, 2013.

[11] V. Kekatos and G. B. Giannakis, "Distributed robust power system state estimation," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1617–1626, 2012.

[12] E. Dall'Anese, H. Zhu, and G. B. Giannakis, "Distributed optimal power flow for smart microgrids," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1464–1475, 2013.

[13] G. Lan, S. Lee, and Y. Zhou, "Communication-efficient algorithms for decentralized and stochastic optimization," *Mathematical Programming*, pp. 1–48, 2017.

[14] B. Huang, L. Liu, H. Zhang, Y. Li, and Q. Sun, "Distributed optimal economic dispatch for microgrids considering communication delays," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 8, pp. 1634–1642, 2019.

[15] Y.-C. Cheng and T. G. Robertazzi, "Distributed computation with communication delay (distributed intelligent sensor networks)," *IEEE transactions on aerospace and electronic systems*, vol. 24, no. 6, pp. 700–712, 1988.

[16] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 1, no. 54, pp. 48–61, 2009.

[17] D. Thanou, E. Kokiopoulou, Y. Pu, and P. Frossard, "Distributed average consensus with quantization refinement," *IEEE Transactions on Signal Processing*, vol. 61, no. 1, pp. 194–205, 2012.

[18] Y. Song, G. Wei, and S. Liu, "Distributed output feedback mpc with randomly occurring actuator saturation and packet loss," *International Journal of Robust and Nonlinear Control*, vol. 26, no. 14, pp. 3036–3057, 2016.

[19] Y. Zhang and Y.-P. Tian, "Consensus of data-sampled multi-agent systems with random communication delay and packet loss," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 939–943, 2010.

[20] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1063–1076, 2018.

[21] L. Su and N. H. Vaidya, "Byzantine-resilient multi-agent optimization," *IEEE Transactions on Automatic Control*, 2020 (Early Access).

[22] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed model predictive control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 44–52, 2002.

[23] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *2011 50th IEEE conference on decision and control and European control conference*, IEEE, 2011, pp. 3800–3805.

[24] A. G. Dimakis, S. Kar, J. M. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.

[25] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 51–60, 2002.

[26] A. Nedic, "Distributed gradient methods for convex machine learning problems in networks: Distributed optimization," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 92–101, 2020.

[27] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[28] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Domputation: Numerical Methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[29] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[30] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[31] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[32] K. I. Tsianos and M. G. Rabbat, "Distributed strongly convex optimization," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2012, pp. 593–600.

[33] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic control*, vol. 57, no. 3, pp. 592–606, 2011.

[34] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[35] S. Pu, W. Shi, J. Xu, and A. Nedic, "Push-pull gradient methods for distributed optimization in networks," *IEEE Transactions on Automatic Control*, 2020.

[36] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[37] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[38] C. Xi and U. A. Khan, "Dextra: A fast algorithm for optimization over directed graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4980–4993, 2017.

[39] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015.

[40] D. Jakovetić, "A unification and generalization of exact distributed first-order methods," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 31–46, 2019.

[41] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[42] T. H. Chang, M. Hong, and X. Wang, "Multi-agent distributed optimization via inexact consensus ADMM," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 482–497, 2014.

[43] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[44] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[45] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "Dqm: Decentralized quadratically approximated alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5158–5173, 2016.

[46] D. Bertsimas and A. Thiele, "Robust and data-driven optimization: Modern decision making under uncertainty," in *Models, methods, and applications for innovative decision making*, INFORMS, 2006, pp. 95–122.

[47] E. Delage and Y. Ye, "Distributionally robust optimization under moment uncertainty with application to data-driven problems," *Operations Research*, vol. 58, no. 3, pp. 595–612, 2010.

[48] C. Zhao and Y. Guan, "Data-driven stochastic unit commitment for integrating wind generation," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 2587–2596, 2015.

[49] R. Jiang and Y. Guan, "Data-driven chance constrained stochastic program," *Mathematical Programming*, vol. 158, no. 1-2, pp. 291–327, 2016.

[50] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[51] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, 10. Springer series in statistics New York, 2001, vol. 1.

[52] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Springer, 2013, vol. 112.

[53] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[54] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[55] G. Lan, "An optimal method for stochastic composite optimization," *Mathematical Programming*, vol. 133, no. 1-2, pp. 365–397, 2012.

[56] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[57] A. Nedic and S. Lee, "On stochastic subgradient mirror-descent algorithm with weighted averaging," *SIAM Journal on Optimization*, vol. 24, no. 1, pp. 84–107, 2014.

[58] S. Ghadimi, G. Lan, and H. Zhang, "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization," *Mathematical Programming*, vol. 155, no. 1-2, pp. 267–305, 2016.

[59] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," in *Advances in Neural Information Processing Systems*, 2011, pp. 1647–1655.

[60] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, "Optimal distributed online prediction using mini-batches," *Journal of Machine Learning Research*, vol. 13, no. Jan, pp. 165–202, 2012.

[61] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems*, 2013, pp. 315–323.

[62] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence _rate for finite training sets," in *Advances in Neural Information Processing Systems*, 2012, pp. 2663–2671.

[63] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 567–599, 2013.

[64] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems*, 2014, pp. 1646–1654.

[65] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[66] S. Shalev-Shwartz and T. Zhang, "Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization," *Mathematical programming*, vol. 155, no. 1-2, pp. 105–145, 2016.

[67] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč, "Mini-batch semi-stochastic gradient descent in the proximal setting," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 2, pp. 242–255, 2015.

[68] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.

[69] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. J. Smola, "On variance reduction in stochastic gradient descent and its asynchronous variants," in *Advances in Neural Information Processing Systems*, 2015, pp. 2647–2655.

[70] R. Leblond, F. Pedregosa, and S. Lacoste-Julien, "ASAGA: Asynchronous parallel saga," in *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 46–54.

[71] W. K. K. Haneveld and M. H. van der Vlerk, "Optimizing electricity distribution using two-stage integer recourse models," in *Stochastic optimization: algorithms and applications*, Springer, 2001, pp. 137–154.

[72] M. A. H. Dempster, M. Fisher, L. Jansen, B. Lageweg, J. K. Lenstra, and A. Rinnooy Kan, "Analytical evaluation of hierarchical planning systems," *Operations Research*, vol. 29, no. 4, pp. 707–716, 1981.

[73] G. Laporte, F. Louveaux, and H. Mercure, "The vehicle routing problem with stochastic travel times," *Transportation science*, vol. 26, no. 3, pp. 161–170, 1992.

[74] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[75] A. Ruszczyński, "Decomposition methods in stochastic programming," *Mathematical programming*, vol. 79, no. 1-3, pp. 333–353, 1997.

[76] S. Uryasev and P. M. Pardalos, *Stochastic Optimization: Algorithms and Applications*. Springer Science & Business Media, 2013, vol. 54.

[77] R. M. Van Slyke and R. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming," *SIAM Journal on Applied Mathematics*, vol. 17, no. 4, pp. 638–663, 1969.

[78] R. T. Rockafellar and R. J.-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty," *Mathematics of Operations Research*, vol. 16, no. 1, pp. 119–147, 1991.

[79] C. C. CarøE and R. Schultz, "Dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 24, no. 1-2, pp. 37–45, 1999.

[80] S. Sen, "Subgradient decomposition and differentiability of the recourse function of a two stage stochastic linear program," *Operations Research Letters*, vol. 13, no. 3, pp. 143–148, 1993.

[81] T. Archibald, C. Buchanan, K. McKinnon, and L. Thomas, "Nested benders decomposition and dynamic programming for reservoir optimisation," *Journal of the Operational Research Society*, vol. 50, no. 5, pp. 468–479, 1999.

[82] L. Ntaimo, "Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse," *Operations Research*, vol. 58, no. 1, pp. 229–243, 2010.

[83] S. M. Ryan, R. J.-B. Wets, D. L. Woodruff, C. Silva-Monroy, and J. P. Watson, "Toward scalable, parallel progressive hedging for stochastic unit commitment," in *2013 IEEE Power & Energy Society General Meeting*, IEEE, 2013, pp. 1–5.

[84] J. Eckstein, J. P. Watson, and D. L. Woodruff, *Asynchronous projective hedging for stochastic programming*, 2018.

[85] V. S. Mai and E. H. Abed, "Distributed optimization over directed graphs with row stochasticity and constraint regularity," *Automatica*, vol. 102, pp. 94–104, 2019.

[86] D. P. Bertsekas and J. N. Tsitsiklis, "Gradient convergence in gradient methods with errors," *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 627–642, 2000.

[87] B. Du, J. Zhou, and D. Sun, "Improving the convergence of distributed gradient descent via inexact average consensus," *Journal of Optimization Theory and Applications*, pp. 1–18, 2020.

[88] A. S. Berahas, R. Bollapragada, N. S. Keskar, and E. Wei, "Balancing communication and computation in distributed optimization," *IEEE Transactions on Automatic Control*, vol. 64, no. 8, pp. 3141–3155, 2018.

[89] S. Sundaram and C. N. Hadjicostis, "Finite-time distributed consensus in graphs with time-invariant topologies," in *2007 American Control Conference*, IEEE, 2007, pp. 711–716.

[90] S. Mou and A. S. Morse, "Finite-time distributed averaging," in *2014 American Control Conference*, IEEE, 2014, pp. 5260–5263.

[91] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," *Lecture notes*, vol. 3, no. 4, p. 5, 1998.

[92] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, 2013.

[93] B. Gerencsér and J. M. Hendrickx, "Push-sum with transmission failures," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1019–1033, 2018.

[94] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," in *45th Annual allerton conf*, 2007, pp. 715–720.

[95] C. Xi, Q. Wu, and U. A. Khan, "On the distributed optimization over directed networks," *Neurocomputing*, vol. 267, pp. 508–515, 2017.

[96] V. S. Mai and E. H. Abed, "Distributed optimization over weighted directed graphs using row stochastic matrix," in *2016 American Control Conference (ACC)*, IEEE, 2016, pp. 7165–7170.

[97] X. Li, G. Feng, and L. Xie, "Distributed proximal algorithms for multi-agent optimization with coupled inequality constraints," *IEEE Transactions on Automatic Control*, 2020.

[98] G. Yu, "Min-max optimization of several classical discrete optimization problems," *Journal of Optimization Theory and Applications*, vol. 98, no. 1, pp. 221–242, 1998.

[99] A. Carè, S. Garatti, and M. C. Campi, "Scenario min-max optimization and the risk of empirical costs," *SIAM Journal on Optimization*, vol. 25, no. 4, pp. 2061–2080, 2015.

[100] H. Aissi, C. Bazgan, and D. Vanderpooten, "Min–max and min–max regret versions of combinatorial optimization problems: A survey," *European Journal of Operational Research*, vol. 197, no. 2, pp. 427–438, 2009.

[101] K. You, R. Tempo, and P. Xie, "Distributed algorithms for robust convex optimization via the scenario approach," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 880–895, 2018.

[102] L. Carlone, V. Srivastava, F. Bullo, and G. C. Calafiore, "Distributed random convex programming via constraints consensus," *SIAM Journal on Control and Optimization*, vol. 52, no. 1, pp. 629–662, 2014.

[103] K. Srivastava, A. Nedić, and D. Stipanović, "Distributed Bregman-distance algorithms for min-max optimization," in *Agent-Based Optimization*, Springer, 2013, pp. 143–174.

[104] I. Notarnicola, M. Franceschelli, and G. Notarstefano, "A duality-based approach for distributed min–max optimization," *IEEE Transactions on Automatic Control*, vol. 64, no. 6, pp. 2559–2566, 2018.

[105] Y. Chen, S. Kar, and J. M. Moura, "Resilient distributed estimation: Sensor attacks," *IEEE Transactions on Automatic Control*, vol. 64, no. 9, pp. 3772–3779, 2018.

[106] Q. Zhou, M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah, "A cyber-attack resilient distributed control strategy in islanded microgrids," *IEEE Transactions on Smart Grid*, vol. 11, no. 5, pp. 3690–3701, 2020.

[107] W. Zeng, Y. Zhang, and M.-Y. Chow, "Resilient distributed energy management subject to unexpected misbehaving generation units," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 208–216, 2015.

[108] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE transactions on robotics*, vol. 33, no. 3, pp. 565–582, 2017.

[109] X. Wang, S. Mou, and S. Sundaram, "A resilient convex combination for consensus-based distributed algorithms," *Numerical Algebra, Control & Optimization*, vol. 9, no. 3, p. 269, 2019.

[110] P. Xie, K. You, R. Tempo, S. Song, and C. Wu, "Distributed convex optimization with inequality constraints over time-varying unbalanced digraphs," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4331–4337, 2018.

[111] J. R. Reay, "An extension of Radon's theorem," *Illinois Journal of Mathematics*, vol. 12, no. 2, pp. 184–189, 1968.

[112] J.-P. Roudneff, "New cases of Reay's conjecture on partitions of points into simplices with k-dimensional intersection," *European Journal of Combinatorics*, vol. 30, no. 8, pp. 1919–1943, 2009.

[113] A. Løkketangen and D. L. Woodruff, "Progressive hedging and tabu search applied to mixed integer (0, 1) multistage stochastic programming," *Journal of Heuristics*, vol. 2, no. 2, pp. 111–128, 1996.

[114] J. P. Watson and D. L. Woodruff, "Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems," *Computational Management Science*, vol. 8, no. 4, pp. 355–370, 2011.

[115] D. Gade, G. Hackebeil, S. M. Ryan, J. P. Watson, R. J.-B. Wets, and D. L. Woodruff, "Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs," *Mathematical Programming*, vol. 157, no. 1, pp. 47–67, 2016.

[116] J. P. Watson, D. L. Woodruff, and W. E. Hart, "PySP: Modeling and solving stochastic programs in python," *Mathematical Programming Computation*, vol. 4, no. 2, pp. 109–149, 2012.

[117] W. E. Hart, C. D. Laird, J. P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo-optimization modeling in python*. Springer, vol. 67.

[118] C. H. Lim, J. T. Linderoth, J. R. Luedtke, and S. J. Wright, "Parallelizing subgradient methods for the Lagrangian dual in stochastic mixed-integer programming," *Informs Journal on Optimization*, vol. 3, no. 1, pp. 1–22, 2021.

[119] I. Aravena and A. Papavasiliou, "A distributed asynchronous algorithm for the two-stage stochastic unit commitment problem," in *2015 IEEE Power & Energy Society General Meeting*, IEEE, 2015, pp. 1–5.

[120] M. Lubin, K. Martin, C. G. Petra, and B. Sandıkçı, "On parallelizing dual decomposition in stochastic integer programming," *Operations Research Letters*, vol. 41, no. 3, pp. 252–258, 2013.

[121] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers Inc, 2011.

[122] Z. Tian, Z. Zhang, J. Yan, and J. Wang, "Distributed ADMM with synergetic communication and computation," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2020, pp. 956–962.

[123] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed ADMM over networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5082–5095, 2017.

[124] Z. Xu, G. Taylor, H. Li, M. A. Figueiredo, X. Yuan, and T. Goldstein, "Adaptive consensus ADMM for distributed optimization," in *International Conference on Machine Learning*, 2017, pp. 3841–3850.

[125] N. Boland, J. Christiansen, B. Dandurand, A. Eberhard, J. Linderoth, J. Luedtke, and F. Oliveira, "Combining progressive hedging with a Frank–Wolfe method to compute Lagrangian dual bounds in stochastic mixed-integer programming," *SIAM Journal on Optimization*, vol. 28, no. 2, pp. 1312–1336, 2018.

[126] N. Z. Shor, *Minimization methods for non-differentiable functions*. Springer Science & Business Media, 2012, vol. 3.

[127] S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, F. Qiu, and S. Sen, "SIPLIB: A stochastic integer programming test problem library," 2015.

[128] S. Jorjani, C. H. Scott, and D. L. Woodruff, "Selection of an optimal subset of sizes," *International Journal of Production Research*, vol. 37, no. 16, pp. 3697–3710, 1999.

[129] P. Wei and D. Sun, "Weighted algebraic connectivity: An application to airport transportation network," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 13 864–13 869, 2011.

[130] P. Wei, G. Spiers, and D. Sun, "Algebraic connectivity maximization for air transportation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 685–698, 2013.

[131] L. Ntaimo and S. Sen, "The million-variable "march" for stochastic combinatorial optimization," *Journal of Global Optimization*, vol. 32, no. 3, pp. 385–400, 2005.

[132] S. d'Oleire-Oltmanns, I. Marzolff, K. Peter, and J. Ries, "Unmanned aerial vehicle (UAV) for monitoring soil erosion in morocco," *Remote Sensing*, vol. 4, no. 11, pp. 3390–3416, 2012.

[133] J. C. Hodgson, S. M. Baylis, R. Mott, A. Herrod, and R. H. Clarke, "Precision wildlife monitoring using unmanned aerial vehicles," *Scientific reports*, vol. 6, p. 22 574, 2016.

[134] A. Goodchild and J. Toy, "Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry," *Transportation Research Part D: Transport and Environment*, vol. 61, pp. 58–67, 2018.

[135] S. Sawadsitang, D. Niyato, P.-S. Tan, and P. Wang, "Joint ground and aerial package delivery services: A stochastic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2241–2254, 2018.

[136] D. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[137] J. J. Acevedo, B. C. Arrue, I. Maza, and A. Ollero, "Cooperative large area surveillance with a team of aerial mobile robots for long endurance missions," *Journal of Intelligent & Robotic Systems*, vol. 70, no. 1-4, pp. 329–345, 2013.

[138] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Science*, vol. 37, no. 6, pp. 351–360, 2006.

[139] B. Du, R. Mao, N. Kong, and D. Sun, "Distributed data fusion for on-scene signal sensing with a multi-uav system," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 3, pp. 1330–1341, 2020.

[140] E. Taghavi, R. Tharmarasa, T. Kirubarajan, Y. Bar-Shalom, and M. Mcdonald, "A practical bias estimation algorithm for multisensor-multitarget tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 1, pp. 2–19, 2016.

[141] X. Lin, Y. Bar-Shalom, and T. Kirubarajan, "Multisensor multitarget bias estimation for general asynchronous sensors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 3, pp. 899–921, 2005.

[142] R. Alonso and M. D. Shuster, "Complete linear attitude-independent magnetometer calibration," *Journal of the Astronautical Sciences*, vol. 50, no. 4, pp. 477–490, 2002.

[143] C. Hajiyev, "In-orbit magnetometer bias and scale factor calibration," *International Journal of Metrology and Quality Engineering*, vol. 7, no. 1, p. 104, 2016.

[144] G. Yang, *Body sensor networks*. Springer, 2006.

[145] A. Grammenos, C. Mascolo, and J. Crowcroft, "You are sensing, but are you biased?: A user unaided sensor calibration approach for mobile sensing," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, p. 11, 2018.

[146] M. Unser and M. Eden, "Weighted averaging of a set of noisy images for maximum signal-to-noise ratio," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 5, pp. 890–895, 1990.

[147] G. H. Golub and C. F. Van Loan, *Matrix Computations*. JHU press, 2012.

[148] F. Penna and S. Stańczak, "Decentralized eigenvalue algorithms for distributed signal detection in wireless networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 427–440, 2015.

[149] J. Ackora Prah, A. Y. Aidoo, and K. B. Gyamfi, "An artificial ECG signal generating function in MATLAB," *Applied Mathematical Sciences*, vol. 7, no. 54, 2013.

[150] B. Du, K. Qian, C. Claudel, and D. Sun, "Multi-robot dynamical source seeking in unknown environments," in *2021 IEEE International Conference on Robotics and Automation*, IEEE, 2021, to appear.

[151] B. Du, K. Qian, C. Claudel, and D. Sun, "Multi-robot dynamical source seeking in unknown environments," *arXiv preprint arXiv:2103.11016*, 2021.

[152] E. Ould-Ahmed-Vall, D. M. Blough, B. H. Ferri, and G. F. Riley, "Distributed global ID assignment for wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 6, pp. 1194–1216, 2009.

[153] B. Du, K. Qian, C. Claudel, and D. Sun, "Jacobi-style iteration for distributed submodular maximization," *arXiv preprint arXiv:2010.14082*, 2020.

[154] W. Li, Z. Wang, D. Ho, and G. Wei, "On boundedness of error covariances for Kalman consensus filtering problems," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2654–2661, 2019.

[155] G. Battistelli and L. Chisci, "Kullback–Leibler average, consensus on probability densities, and distributed state estimation with guaranteed stability," *Automatica*, vol. 50, no. 3, pp. 707–718, 2014.

[156] G. Battistelli, L. Chisci, G. Mugnai, A. Farina, and A. Graziano, "Consensus-based linear and nonlinear filtering," *IEEE Transactions on Automatic Control*, vol. 60, no. 5, pp. 1410–1415, 2014.

[157] F. S. Cattivelli and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering and smoothing," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2069–2084, 2010.

[158] E. Rolf, D. Fridovich-Keil, M. Simchowitz, B. Recht, and C. Tomlin, "A successive-elimination approach to adaptive robotic source seeking," *IEEE Transactions on Robotics*, 2020.

[159] J. Habibi, H. Mahboubi, and A. G. Aghdam, "A gradient-based coverage optimization strategy for mobile sensor networks," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 3, pp. 477–488, 2016.

[160] R. Olfati-Saber, "Distributed Kalman filter with embedded consensus filters," in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 8179–8184.

[161] R. Olfati-Saber and J. Shamma, "Consensus filters for sensor networks and distributed sensor fusion," in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 6698–6703.

[162] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proceedings of the 46th IEEE Conference on Decision and Control*, IEEE, 2007, pp. 5492–5498.

[163] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," *Advances in Neural Information Processing Systems*, vol. 24, pp. 2312–2320, 2011.

[164] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[165] N. Atanasov, R. Tron, V. M. Preciado, and G. J. Pappas, "Joint estimation and localization in sensor networks," in *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 6875–6882.

[166] V. M. H. Bennetts, A. J. Lilienthal, A. A. Khaliq, V. P. Sese, and M. Trincavelli, "Towards real-world gas distribution mapping and leak localization using a mobile robot with 3d and remote gas sensing capabilities," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 2335–2340.

VITA

Bin Du was born in Nanjing, Jiangsu Province, China. He received his Bachelor degree in Electrical Engineering in 2014 from Nanjing University of Aeronautics and Astronautics, Nanjing, China, and his Master degree in Automation Science and Engineering in 2017 from University of Science and Technology of China, Hefei, China. Then, he entered Purdue University, West Lafayette, United States, pursuing his Ph.D. degree in School of Aeronautics and Astronautics. His research interests include distributed optimization and control, with applications in aerospace engineering such as air control, autonomous vehicle, etc.