Privacy-Preserving Decentralized Optimization and Event Localization

Chunlei Zhang
Clemson University, zhangchunlei0813@gmail.com

Privacy-preserving Decentralized Optimization and Event Localization

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

---

by
Chunlei Zhang
December 2019

Abstract

This dissertation considers decentralized optimization and its applications. On the one hand, we address privacy preservation for decentralized optimization, where N agents cooperatively minimize the sum of N convex functions private to these individual agents. In most existing decentralized optimization approaches, participating agents exchange and disclose states explicitly, which may not be desirable when the states contain sensitive information of individual agents. The problem is more acute when adversaries exist which try to steal information from other participating agents. To address this issue, we first propose two privacy-preserving decentralized optimization approaches based on ADMM (alternating direction method of multipliers) and subgradient method, respectively, by leveraging partially homomorphic cryptography. To our knowledge, this is the first time that cryptographic techniques are incorporated in a fully decentralized setting to enable privacy preservation in decentralized optimization in the absence of any third party or aggregator. To facilitate the incorporation of encryption in a fully decentralized manner, we also introduce a new ADMM which allows time-varying penalty matrices and rigorously prove that it has a convergence rate of $O(1/t)$. However, given that encryption-based algorithms unavoidably bring about extra computational and communication overhead in real-time optimization [61], we then propose another novel privacy solution for decentralized optimization based on function decomposition and ADMM which enables privacy without incurring large communication/computational overhead.

On the other hand, we address the application of decentralized optimization to the event localization problem, which plays a fundamental role in many wireless sensor network applications such as environmental monitoring, homeland security, medical treatment, and health care. The event localization problem is essentially a non-convex and non-smooth problem. We address such a problem in two ways. First, a completely decentralized solution based on augmented Lagrangian methods and ADMM is proposed to solve the non-smooth and non-convex problem directly, rather than using conventional convex relaxation techniques. However, this algorithm requires the target event to be within the convex hull of the deployed sensors. To address this issue, we propose another two scalable distributed algorithms based on ADMM and convex relaxation, which do not require the target event to be within the convex hull of the deployed sensors. Simulation results confirm effectiveness of the proposed algorithms.

Acknowledgments

I want to start by sincerely thanking my advisor, Dr. Yongqiang Wang, for all the continuous support, advice, and patience to my Ph.D. study and research at Clemson University. His immense knowledge and guidance helped me get through challenges and difficulties both in my research and life. His contributions of time and novel ideas make my Ph.D. experience productive and stimulating. Under the supervision of Dr. Wang, I am fortunate to build up critical thinking in research and gain rigorous attitude in study. None of the accomplishments in my Ph.D. journey would have been possible without his support. Besides my advisor, I would like to thank for my committee members: Dr. Richard E. Groff, Dr. Yingjie Lao, and Dr. Yuyuan Ouyang, for their time, insightful comments, and valuable feedbacks on my comprehensive exam and dissertation. I also want to express my sincere gratitude to my parents and family members. They always stand by me, providing me with their warm encouragement, continued care, and endless love. They raised me to be a friendly, healthy, and happy person. Another guy I want to say thank you is my boyfriend who is always with me, giving me warm care and support in my life. My sincere thanks also goes to my friends, lab mates and roommates: Wei Li, Xuejiao Yang, Huan Gao, Zhenqian Wang, Yin Li, Xinyuan Ma, Zhanrui Liao, among many others. Whenever I was worried or not in a good mood, they lift me up. I would like to appreciate all of your support and encouragement throughout my Ph.D. life!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Intensive applications and problems in statistics, machine leaning, multi-agent systems, and power grids can be modeled in the framework of optimization. Some examples include the sparse linear regression [77], event localization [78], wide-area oscillation monitoring [83], etc. Although these problems arise in various domains, they share some universal characteristics such as data being collected or stored in different local agents. Given this distributed manner, it is natural to look to decentralized optimization in which data are processed in a decentralized and cooperative manner.

This dissertation addresses decentralized optimization and its application to event localization. In recent years, numerous algorithms were proposed for decentralized optimization such as distributed (sub)gradient based algorithms [69,86,87], augmented Lagrangian methods (ALM) [44,47], and the alternating direction method of multipliers (ADMM) as well as its variants [16,44,47,64,65], etc. However, most of these approaches require agents to exchange and disclose their states explicitly to neighboring agents in every iteration, which may not be desirable when the estimates contain sensitive information of individual agents. To address this privacy issue, this dissertation will first develop privacy-preserving decentralized optimization algorithms. Then, this dissertation will address the application of decentralized optimization to the event localization problem, which plays a fundamental role in many wireless sensor network applications such as environmental monitoring [15], target tracking [21], underwater detection [50], and acoustic gunfire localization [34], [105]. The following two sections will introduce the two topics in more depth.

1.1 Decentralized optimization

The problem of decentralized optimization has attracted remarkable attention in recent years due to its wide applications in various domains, ranging from multi-agent systems [69, 86, 87], machine learning [23, 123, 134], statistics [67, 77], communications and networking [69, 78, 86, 87], to power grids [35, 83]. In these applications, data are collected and processed in a decentralized and cooperative manner among multiple agents. Such a decentralized manner of processing provides several key advantages over its conventional computing-server based counterpart. For example, it leads to enhanced scalability and flexibility, and brings higher robustness to the problems of network traffic bottleneck and single point of failure.

Typical decentralized optimization algorithms include distributed (sub)gradient based algorithms [69,86,87], augmented Lagrangian methods (ALM) [44,47], and the alternating direction method of multipliers (ADMM) as well as its variants [16, 44, 47, 64, 65], etc. In (sub)gradient based solutions, (sub)gradient computations and averaging among neighbors are conducted iteratively to achieve convergence to the minimum. In augmented Lagrangian and ADMM based solutions, iterative Lagrangian minimization is employed, which, coupled with dual variable update, guarantees that all agents agree on the minimization solution.

However, most of the aforementioned decentralized approaches require agents to exchange and disclose their states explicitly to neighboring agents in every iteration [16,47,64,65,86]. This brings about serious privacy concerns in many practical applications [61]. For example, in projection based source localization, intermediate states are positions of points lying on the circles centered at individual nodes' positions [101], and thus a node may infer the exact position of a neighboring node using three intermediate states, which is undesirable when agents want to keep their position private [4]. In the rendezvous problem where a group of individuals want to meet at an agreed time and place [63], exchanging explicit states may leak their initial locations which may need to be kept secret instead [80]. Other examples include the agreement problem [24], where a group of individuals want to reach consensus on a subject without leaking their individual opinions to others [80], and the regression problem [77], where individual agent's training data may contain sensitive information (e.g., salary, medical record) and should be kept private. In addition, exchanging explicit states without encryption is susceptible to eavesdroppers which try to intercept and steal information from exchanged messages.

To enable privacy preservation in decentralized optimization, one commonly used approach is differential privacy [42, 54, 89], which adds carefully-designed noise to exchanged states or objective functions to cover sensitive information. However, the added noise also unavoidably compromises the accuracy of optimization results, leading to a trade-off between privacy and accuracy [42, 54, 89]. In fact, as indicated in [89], even when no noise perturbation is added, differential-privacy based approaches may fail to converge to the accurate optimal solution. It is worth noting that although some differential-privacy based optimization approaches can converge to the optimal solution in the mean-square sense with the assistance of a third party such as a cloud (e.g., [38], [39]), those results are not applicable to the completely decentralized setting discussed here where no third parties or aggregators exist. Observability-based design has been proposed for privacy preservation in linear multi-agent networks [5, 96]. By properly designing the weights for the communication graph, agents' information will not be revealed to non-neighboring agents. However, this approach cannot protect the privacy of the direct neighbors of compromised agents and it is susceptible to external eavesdroppers. Another approach to enabling privacy preservation is encryption. However, despite successful applications in cloud based control and optimization [29, 104, 111, 122], conventional cryptographic techniques cannot be applied directly in a completely *decentralized* setting without the assistance of aggregators/third parties (note that traditional secure multi-party computation schemes like fully homomorphic encryption [33] and Yao's garbled circuit [125] are computationally too heavy to be practical for real-time optimization [61]). Other privacy-preserving optimization approaches include [31, 74] which protect privacy via perturbing problems or states.

To enable privacy preservation without compromising the optimality of the solution, this dissertation first proposes a novel approach that enables privacy-preservation in decentralized optimization through incorporating partially homomorphic cryptography in existing optimization algorithms. We show that cryptographic techniques can be incorporated in a fully decentralized manner to enable privacy-preservation in decentralized optimization in the absence of any third party or aggregator. This is significant in that, to our knowledge, all existing cryptographic based optimization approaches rely on the assistance of a third-party or aggregator to protect the privacy of all parties. However, given that encryption unavoidably brings about significant extra computational and communication overhead in real-time optimization [61], this dissertation also proposes a novel privacy solution for decentralized optimization based on function decomposition which enables privacy without incurring large communication/computational overhead. Our contributions can be summarized as follows:

1. A privacy-preserving decentralized optimization approach is proposed based on ADMM and partially homomorphic cryptography. To facilitate the incorporation of homomorphic encryption in ADMM in a fully decentralized manner, we also propose a new ADMM which allows time-varying penalty matrices and rigorously characterize its convergence rate of $O(1/t)$.

2. A privacy-preserving decentralized optimization approach based on subgradient method and partially homomorphic cryptography is proposed, which can be used in time-varying networks.

3. A novel privacy-preserving decentralized optimization approach is proposed based on ADMM and function decomposition. Compared with encryption-based approaches which suffer from heavy computational and communication burden, the proposed approach incurs little extra computational and communication overhead; In addition, we prove that when the global objective function is strongly convex, proximal Jacobian ADMM can achieve Q-linear convergence rate even if local objective functions are only convex, which generalizes existing results on proximal Jacobian ADMM requiring strongly convex local objective functions to achieve Q-linear convergence rate.

4. In contrast to differential-privacy based optimization approaches [42, 54, 89], our work can enable privacy preservation without sacrificing accuracy.

1.2 Event localization

With the ability to transmit/receive information and fuse data, smart sensors enabled and greatly advanced numerous applications such as environmental monitoring [15], target tracking [21], underwater detection [50], and acoustic gunfire localization [34], [105]. Among these applications, *event localization* is a significant and essential component or even the ultimate goal. Taking the gunfire localization as an example, if some threat sources or impulsive events (e.g., shooting or explosion) occur, it is of imperative importance to localize these threat sources to make prompt reactions (e.g., giving warning, providing aid). A typical example is the PinPoint$^{\text{TM}}$ mobile acoustic localization sensor network [18, 25], which provides the capability for detection and localization of impulsive threat events on battlefields. In fact, sensor network based event localization has received significant attentions and plenty of techniques have been proposed in the literature, using either angle-of-arrival measurements [32, 57, 100], time-of-arrival (ToA) (including time-difference-of-arrival, i.e., TDoA) measurements [51, 124], or received signal strength (RSS) [13, 79, 101, 113–115, 126, 135]. There are also some work that discussed the event localization problem based on noisy range measurements directly, which can be obtained based on ToA, TDoA, or RSS information [10, 30, 56, 91, 92]. Generally speaking, these existing methods for event localization formulate the localization problem as a maximum likelihood estimation problem [91] or a least squares problem [126], which is solved by minimizing the non-convex objective function iteratively [13] or by applying various convex relaxations [124].

From the implementation point of view, existing event localization algorithms can be cast into two categories: centralized approaches and distributed approaches. Centralized approaches always gather (noisy) measurements (e.g., range measurements) obtained by all sensors to a processing center, which then estimates the event location using a certain centralized optimization algorithm. Typical centralized methods include the parallel projection method [56], convex relaxation plus semidefinite programming (SDP) or second-order cone programming method [30, 91, 92, 113, 114, 124, 126]. However, a severe shortcoming of centralized localization algorithms is that the computation complexity at the processing center might be quite high which poses great challenges for low-cost sensor nodes with limited computational capabilities. In addition, the required communication to collect all measurements to a single central node may be problematic due to possible traffic bottleneck and severe constraints on communication ranges. Moreover, once the central node fails due to, e.g., attacks or power depletion, the entire network slips into a state of paralysis. Therefore, techniques solving the event localization problem in a distributed way are crucial for sensor network based event localization.

In contrast to centralized algorithms, distributed localization algorithms are designed to run the computation over the entire network instead of on a processing center. In general, distributed algorithms are often established on massive parallelism or sequential calculations and mutual collaboration [9]. So compared with centralized algorithms, distributed designs have better scalability, flexibility, and failure resilience. One typical distributed approach for event localization is projection-based algorithms. For example, Blatt and coauthors in [13] proposed a projection-onto-convex-sets (POCS) method which solves the event localization problem via projecting an initial estimate to sensing disks that center at individual sensors' positions. The authors in [101] proposed a nearest local minimum (PONLM) method that projects initial estimates to sensing circles rather than disks, which improved the performance of event localization. The authors in [115] proposed a boundary-of-convex-sets algorithm, with convergence guaranteed in the case of two anchor sensors. Wang and coauthors [112] recently proposed a recursive weighted least squares (RWLS) algorithm which takes information reliability into account by adding weighting factors of the previous estimates in each iteration. However, these projection-based algorithms update local estimates sequentially, which requires a global updating order and hence is not amenable for parallelization. The sequential nature entails a reschedule of global updating order whenever the network topology changes, due to e.g., a sensor's joining or leaving the network, and hence is not as flexible as parallel algorithms. To address this issue, Zhang and coauthors [135] proposed a parallel distributed alternating projection algorithm (DAPA) which formulates the event localization problem as a ring intersection problem. However, this approach does not work well when the target event lies outside the convex hull of sensors.

In this dissertation, we address the event localization problem by applying the alternating direction method of multipliers [16], which has been proven extremely suitable in distributed convex optimization and some non-convex problems [73, 107, 120]. We used ADMM because it has several advantages. First, ADMM has a fast convergence speed in both primal and dual iterations [64]. By incorporating a quadratic regularization term, ADMM has been shown to be able to obtain satisfactory convergence speed even in ill-conditioned dual functions [65]. Secondly, from the implementation point of view, not only is ADMM easy to parallelize and implement, but it is also robust to noise and computation errors [106]. Our proposed algorithms takes full advantages of ADMM which decomposes a general optimization problem into multiple local optimization subproblems with each subproblem solved by an individual part. Through cooperations in the computation process among neighboring sensors, a consistent estimate of the event position across the entire network can be achieved. Our main contributions for the event localization problem are summarized as follows:

1. An algorithm is applied to directly solve the general non-smooth and non-convex event localization problem without using convex relaxation. The avoidance of convex relaxation is significant in that convex relaxation based methods generally suffer from high computational complexity. The proposed algorithm takes full advantages of alternating direction method of multipliers and accomplishes the event localization in a decentralized way. Therefore, compared with centralized approach in which a processing center performs the whole heavy computation, the algorithm is highly scalable, flexible, robust to network topology changes, and thus is more favorable in practical implement;

2. Two distributed event localization algorithms based on ADMM and convex relaxation are proposed. Compared with existing centralized SDP relaxation based algorithms for event localization, the two algorithms divide the computation on a central node to different clusters to avoid possible center failure and traffic bottleneck, and in the mean time, guarantee consistency of the estimates across all clusters among which only limited communications are available. Furthermore, the two algorithms take advantages of SDP relaxation to avoid the convex hull problem compared with existing projection-based algorithms. Moreover, the algorithms are proven to converge with a convergence rate of $O(1/t)$ where t is the iteration time.

The rest of this dissertation is organized as follows: Chapter 2 gives more details on two typical decentralized optimization algorithms and the homomorphic Paillier cryptosystem, which is used to enable privacy-preservation in decentralized optimization. Chapter 3 and Chapter 4 propose a privacy-preserving decentralized optimization algorithm based on ADMM and subgradient method, respectively, by leveraging partially homomorphic cryptography. Chapter 5 proposes a novel privacy-preserving decentralized optimization algorithm based on ADMM and function decomposition. Chapter 6 and Chapter 7 address the application of ADMM to the event localization problem. Finally, we conclude our findings in Chapter 8.

It is worth noting that this dissertation interpolates material from four papers by the author [129–132]. Chapter 3 uses materials from Ref [129], Chapter 4 uses materials from Ref [131], Chapter 6 uses materials from Ref [132], and Chapter 7 uses materials from Ref [130].

Chapter 2

Background and Preliminaries

2.1 Typical Decentralized Optimization Algorithms

2.1.1 Alternating Direction Method of Multipliers

ADMM is an algorithm which is suitable to solve problems in the following form [16]:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & C\boldsymbol{x} + F\boldsymbol{z} = \boldsymbol{c}. \end{aligned} \tag{2.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{z} \in \mathbb{R}^m$, $C \in \mathbb{R}^{p\times n}$, $F \in \mathbb{R}^{p\times m}$, $\boldsymbol{c} \in \mathbb{R}^p$, and $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ are convex functions. To get the optimal value $p^* = \inf\{f(\boldsymbol{x}) + g(\boldsymbol{z}) \mid C\boldsymbol{x} + F\boldsymbol{z} = \boldsymbol{c}\}$ for problem (2.1), one can first form an augmented Lagrangian function:

$$\mathcal{L}_\rho(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^T(C\boldsymbol{x} + F\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2} \parallel C\boldsymbol{x} + F\boldsymbol{z} - \boldsymbol{c} \parallel^2,$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier associated with the constraint $C\boldsymbol{x} + F\boldsymbol{z} = \boldsymbol{c}$ and $\rho > 0$ is a predefined penalty parameter. Then ADMM solves problem (2.1) by updating $\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}$ in the following sequence: first an

$\boldsymbol{x}$-minimization step (2.2), then a $\boldsymbol{z}$-minimization step (2.3), and finally a dual variable update (2.4):

$$\boldsymbol{x}^{k+1} = \text{argmin}_{\boldsymbol{x}} \mathcal{L}_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k), \tag{2.2}$$

$$\boldsymbol{z}^{k+1} = \text{argmin}_{\boldsymbol{z}} \mathcal{L}_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}^k), \tag{2.3}$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho(C\boldsymbol{x}^{k+1} + F\boldsymbol{z}^{k+1} - \boldsymbol{c}). \tag{2.4}$$

Under the assumptions that both f and g are closed, proper, convex, and the Lagrange function $L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^T(C\boldsymbol{x} + F\boldsymbol{z} - \boldsymbol{c})$ has a saddle point, ADMM has primary residual convergence, i.e., $C\boldsymbol{x}^k + F\boldsymbol{z}^k - \boldsymbol{c} \to 0$, objective convergence, i.e., $p^k = f(\boldsymbol{x}^k) + g(\boldsymbol{z}^k) \to p^*$, and dual residual convergence, i.e., $\rho C^T F(\boldsymbol{z}^k - \boldsymbol{z}^{k-1}) \to 0$ when $k \to \infty$. Variations and extensions to standard ADMM algorithm can refer to proximal ADMM [26], linearized ADMM [66], weighted ADMM [64], etc.

2.1.2 Subgradient Method

Given a function $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$. To minimize f, the subgradient method takes the following iteration [17]:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \alpha_k d^k.$$

Here $\boldsymbol{x}^k$ is the kth iterate, $\alpha_k > 0$ is the step size, and d^k is a subgradient of f at $\boldsymbol{x}^k$. The subgradient method is very simple. Under the assumptions that the step size α_k is diminishing ($\lim\limits_{k\to\infty} \alpha_k = 0, \sum_{k=1}^{\infty} \alpha_k = \infty$) and function f is convex, the subgradient method is guaranteed to converge to the optimal value [17].

A typical extension to subgradient method is projected subgradient method which solves a constrained convex optimization problem as follows:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X} \end{aligned}$$

where $\mathcal{X}$ is a convex set. The projected subgradient method takes the following iteration [17]:

$$\boldsymbol{x}^{k+1} = P_{\mathcal{X}}(\boldsymbol{x}^k - \alpha_k d^k).$$

Here $P_{\mathcal{X}}[\cdot]$ denotes the projection operation onto the set $\mathcal{X}$, i.e., $P_{\mathcal{X}}[\boldsymbol{r}] = \operatorname*{argmin}_{\boldsymbol{y}\in\mathcal{X}} \parallel \boldsymbol{y} - \boldsymbol{r} \parallel$. In addition, when the objective function f is a sum of $N \geq 2$ individual functions, the (projected) subgradient method can be easily implemented in a decentralized manner (c.f. [86, 87]).

2.2 Paillier Cryptosystem

The Paillier cryptosystem is a public-key cryptosystem which uses a pair of keys: a public key and a private key. The public key can be disseminated publicly and used by any person to encrypt a message, but the message can only be decrypted by the private key. The Paillier cryptosystem includes three algorithms, which are detailed below:

Paillier cryptosystem

Key generation:

1. Choose two large prime numbers p and q of equal bit-length and compute $n = pq$.
2. Let $g = n + 1$.
3. Let $\lambda = \phi(n) = (p-1)(q-1)$, where $\phi(\cdot)$ is the Euler's totient function.
4. Let $\mu = \phi(n)^{-1} \mod n$ which is the modular multiplicative inverse of $\phi(n)$.
5. The public key k_p for encryption is (n, g).
6. The private key k_s for decryption is (λ, μ).

Encryption ($c = \mathcal{E}(m)$): Recall the definitions of $\mathbb{Z}_n = \{z | z \in \mathbb{Z}, 0 \leq z < n\}$ and $\mathbb{Z}_n^* = \{z | z \in \mathbb{Z}, 0 \leq z < n, \gcd(z, n) = 1\}$.

1. Choose a random $r \in \mathbb{Z}_n^*$.
2. The ciphertext is given by $c = g^m \cdot r^n \mod n^2$, where $m \in \mathbb{Z}_n, c \in \mathbb{Z}_{n^2}^*$.

Decryption ($m = D(c)$):

1. Define the integer division function $L(\mu) = \frac{\mu - 1}{n}$.
2. The plaintext is $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

A notable feature of Paillier cryptosystem is that it is additively homomorphic, i.e., the ciphertext of $m_1 + m_2$ can be obtained from the ciphertext of m_1 and m_2 directly when $0 \leq m_1 + m_2 < n$ holds:

$$\mathcal{E}(m_1, r_1) \cdot \mathcal{E}(m_2, r_2) = \mathcal{E}(m_1 + m_2, r_1 r_2), \tag{2.5}$$

$$\mathcal{E}(m)^k = \mathcal{E}(km), \quad k \in \mathbb{Z}^+. \tag{2.6}$$

Due to the existence of random r, the Paillier cryptosystem is resistant to the dictionary attack [36]. Since r_1

and r_2 play no role in the decryption process, (2.5) can be simplified as

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = \mathcal{E}(m_1 + m_2). \tag{2.7}$$

Chapter 3

Privacy-preserving Decentralized Optimization Based on ADMM

3.1 Introduction

An important class of decentralized optimization problems is to minimize an objective function that is the sum of N convex functions private to N individual agents [54, 69, 86, 87]. Such decentralized optimization has been playing key roles in applications as diverse as rendezvous in multi-agent systems [63], spectrum sensing in cognitive networks [128], support vector machine [23] and classification [134] in machine learning, online learning [123], data regression in statistics [77], source localization in sensor networks [130], and monitoring of smart grids [83]. This chapter considers such a decentralized problem in which N agents cooperatively solve an unconstrained optimization:

$$\min_{\tilde{\boldsymbol{x}}} \qquad \sum_{i=1}^{N} f_i(\tilde{\boldsymbol{x}}), \tag{3.1}$$

where variable $\tilde{\boldsymbol{x}} \in \mathbb{R}^D$ is common to all agents, function $f_i : \mathbb{R}^D \to \mathbb{R}$ is the local objective function of agent i. We propose a new privacy-preserving decentralized optimization approach based on ADMM and partially homomorphic cryptography to solve (3.1) in this chapter. To facilitate the incorporation of homomorphic encryption in ADMM in a fully decentralized manner, we also propose a new ADMM which allows time-varying penalty matrices and rigorously characterize its convergence rate of $O(1/t)$.

It is worth noting that privacy has different meanings under different settings. For example, in the distributed optimization literature, privacy has been defined as the non-disclosure of agents' states [38], objective functions or subgradients [71, 89, 123]. In this chapter, we define privacy as preserving the confidentiality of agents' intermediate states, gradients of objective functions, and objective functions. We protect the privacy of objective functions through protecting intermediate states. In fact, if left unprotected, intermediate states could be used by an adversary to infer the gradients or even objective functions of other nodes through, e.g., data mining techniques. For example, in the regression problem in [77], the objective functions take the form $f_i(\tilde{\boldsymbol{x}}) = \frac{1}{2} \parallel \boldsymbol{s}_i - B_i\tilde{\boldsymbol{x}} \parallel_2^2$, in which s_i and B_i are raw data containing sensitive information such as salary and medical record. When the subgradient method in [86] is used to solve the optimization problem $\min\limits_{\tilde{\boldsymbol{x}}} \sum\limits_{i=1}^{N} f_i(\tilde{\boldsymbol{x}})$, agent i updates its intermediate states in the following way:

$$\boldsymbol{x}_i^{k+1} = \sum_{j=1}^{N} a_{ij}\boldsymbol{x}_j^k - \alpha_k \nabla f_i(\boldsymbol{x}_i^k)$$

where a_{ij} are weights, α_k is the stepsize, and $\nabla f_i(\boldsymbol{x}) = B_i^T B_i \boldsymbol{x} - B_i^T \boldsymbol{s}_i$ is the gradient. In this case, an adversary can infer $\nabla f_i(\boldsymbol{x}_i^k)$ based on exchanged intermediate states $\boldsymbol{x}_i$ if the weights a_{ij} and stepsize α_k are publicly known. We consider two adversaries in this chapter: *Honest-but-curious adversaries* are agents who follow all protocol steps correctly but are curious and collect all intermediate and input/output data in an attempt to learn some information about other participating agents [62]. *External eavesdroppers* are adversaries who steal information through wiretapping all communication channels and intercepting exchanged messages between agents. Protecting agents' intermediate states can avoid eavesdroppers from inferring any information in optimization.

Organization: The rest of this chapter is organized as follows: Sec. 3.2 first presents the conventional ADMM solution to (3.1), and then introduces a new ADMM which allows time-varying penalty matrices with guaranteed convergence. Based on the new ADMM and partially homomorphic cryptography, a completely decentralized privacy-preserving approach to solving problem (3.1) is proposed in Sec. 3.3. Rigorous analysis of the guaranteed privacy of the approach are addressed in Sec. 3.4 and its implementation details are discussed in Sec. 3.5. Numerical simulation results are given in Sec. 3.6 and Sec. 3.7 to confirm the effectiveness and computational efficiency of the proposed approach. In the end, we draw summaries in Sec. 3.8.

3.2 A New ADMM with Time-varying Penalty Matrices

In this section, we propose a new ADMM with time-varying penalty matrices for (3.1), which is key for enabling the incorporation of partially homomorphic cryptography in a completely decentralized optimization problem for privacy protection.

3.2.1 Problem Formulation

We assume that each f_i in (3.1) is private and only known to agent i, and all N agents form a bidirectional connected network. Using the graph theory [14], we represent the communication pattern of a multi-agent network by a graph $G = \{V, E\}$, where V denotes the set of agents and E denotes the set of communication links (undirected edges) between agents. Denote the total number of communication links in E as $|E|$. If there exists a communication link between agents i and j, we say that agent j is a neighbor of i (agent i is a neighbor of j as well) and denote the communication link as $e_{i,j} \in E$ if $i < j$ is true or $e_{j,i} \in E$ if $i > j$ is true. Moreover, we denote the set of all neighboring agents of i as $\mathcal{N}_i$ (we consider agent i to be a neighbor of itself in this chapter, i.e., $i \in \mathcal{N}_i$, but $e_{i,i} \notin E$).

3.2.2 Proximal Jacobian ADMM

To solve (3.1) in a decentralized manner, we reformulate (3.1) as follows (which avoids using dummy variables in conventional ADMM [103]):

$$\begin{aligned} \min_{\boldsymbol{x}_i \in \mathbb{R}^D,\, i \in \{1,2,\ldots,N\}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{x}_j, \quad \forall e_{i,j} \in E, \end{aligned} \tag{3.2}$$

where $\boldsymbol{x}_i$ is considered as a copy of $\boldsymbol{x}$ and belongs to agent i. To solve (3.2), each agent first exchanges its current state $\boldsymbol{x}_i$ with its neighbors. Then it carries out local computations based on its private local objective function f_i and the received state information from neighbors to update its state. Iterating these computations will make every agent reach consensus on a solution that is optimal to (3.1) when (3.1) is convex. Detailed implementation of the ADMM algorithm based on Jacobian update is elaborated as follows [26]:

$$\begin{cases} \boldsymbol{x}_i^{t+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, ..., \boldsymbol{x}_{i-1}^t, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^t, ..., \boldsymbol{x}_N^t, \boldsymbol{\lambda}^t) + \frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2, & (3.3) \\ \boldsymbol{\lambda}_{i,j}^{t+1} = \lambda_{i,j}^t + \rho(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}), \quad \forall j \in \mathcal{N}_i & (3.4) \end{cases}$$

for $i = 1, 2, ..., N$. Here, t is the iteration index, $\gamma_i > 0$ $(i = 1, 2, ..., N)$ are proximal coefficients, and $\mathcal{L}$ is the augmented Lagrangian function

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + \sum_{e_{i,j} \in E} (\boldsymbol{\lambda}_{i,j}^T(\boldsymbol{x}_i - \boldsymbol{x}_j) + \frac{\rho}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2). \tag{3.5}$$

In (3.5), $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_N^T]^T \in \mathbb{R}^{ND}$ is the augmented state, $\boldsymbol{\lambda}_{i,j}$ is the Lagrange multiplier corresponding to the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j$, and all $\boldsymbol{\lambda}_{i,j}$ for $e_{i,j} \in E$ are stacked into $\boldsymbol{\lambda} \in \mathbb{R}^{|E|D}$. ρ is the penalty parameter, which is a positive constant scalar.

The above ADMM algorithm cannot protect the privacy of participating agents as states are exchanged and disclosed explicitly among neighboring agents. To facilitate privacy design, we propose a new ADMM with time-varying penalty matrices in the following subsection, which will enable the integration of homomorphic cryptography and decentralized optimization in Sec. 3.3.

3.2.3 ADMM with Time-varying Penalty Matrices

Motivated by the fact that ADMM allows time-varying penalty matrices [45, 58], we present in the following an ADMM with time-varying penalty matrices. It is worth noting that [45, 58] deal with a two-block $(N = 2)$ problem. While in this chapter, we consider a more general problem with $N \geq 3$ blocks, whose convergence is more difficult to analyze. The generalization from $N = 2$ to $N \geq 3$ is highly non-trivial. In fact, as indicated in [22], a direct extension from two-block to multi-block convex minimization is not necessarily convergent.

We first reformulate (3.1) in a more compact form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & A\boldsymbol{x} = \boldsymbol{0}, \end{aligned} \tag{3.6}$$

where $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_N^T]^T \in \mathbb{R}^{ND}$, $f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i)$, and $A = [a_{m,n}] \otimes I_D \in \mathbb{R}^{|E|D \times ND}$ is the edge-node incidence matrix of graph G as defined in [118], with its $|E|D$ rows corresponding to the $|E|$ communication links and the ND columns corresponding to the N agents. The symbol $\otimes$ denotes Kronecker product. The $a_{m,n}$ element is defined as

$$a_{m,n} = \begin{cases} 1 & \text{if the } m^{th} \text{ edge originates from agent } n, \\ -1 & \text{if the } m^{th} \text{ edge terminates at agent } n, \\ 0 & \text{otherwise.} \end{cases}$$

Here we define that each edge $e_{i,j}$ originates from agent i and terminates at agent j.

Let $\boldsymbol{\lambda}_{i,j}$ be the Lagrange multiplier corresponding to the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j$, then we can form an augmented Lagrangian function of problem (3.6) as

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\rho}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + \sum_{e_{i,j} \in E} (\boldsymbol{\lambda}_{i,j}^T(\boldsymbol{x}_i - \boldsymbol{x}_j) + \frac{\rho_{i,j}}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2), \tag{3.7}$$

or in a more compact form:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\rho}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x} + \frac{1}{2} \parallel A\boldsymbol{x} \parallel_{\boldsymbol{\rho}}^2, \tag{3.8}$$

where $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_{i,j}]_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D}$ is the augmented Lagrange multiplier,

$$\boldsymbol{\rho} = \text{diag}\{\rho_{i,j} I_D\}_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D \times |E|D}, \quad \rho_{i,j} > 0$$

is the time-varying penalty matrix, and $\parallel A\boldsymbol{x} \parallel_{\boldsymbol{\rho}}^2 = \boldsymbol{x}^T A^T \boldsymbol{\rho} A\boldsymbol{x}$.

Note that if $\rho_{i,j} I_D$ is the mth block in $\boldsymbol{\rho}$, then $e_{i,j}$ is the mth edge in E, i.e., for the one-dimensional case, $a_{m,i} = 1$ and $a_{m,j} = -1$, and for high dimensional cases, the (m, i)th block of A is I_D and the (m, j)th block of A is $-I_D$.

Now, inspired by [45], we propose a new ADMM which allows time-varying penalty matrices based on Jacobian update [99]:

$$\begin{cases} \boldsymbol{x}_i^{t+1} = \underset{\boldsymbol{x}_i}{\text{argmin}} \mathcal{L}(\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, ..., \boldsymbol{x}_{i-1}^t, \boldsymbol{x}_i, \boldsymbol{x}_{i+1}^t, ..., \boldsymbol{x}_N^t, \boldsymbol{\lambda}^t, \boldsymbol{\rho}^t) + \frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2, & (3.9) \\ \rho_{i,j}^t \to \rho_{i,j}^{t+1}, & (3.10) \\ \boldsymbol{\lambda}_{i,j}^{t+1} = \lambda_{i,j}^t + \rho_{i,j}^{t+1}(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}), \quad \forall j \in \mathcal{N}_i & (3.11) \end{cases}$$

for $i = 1, 2, ..., N$. It is worth noting that although the communication graph is undirected, we introduce both $\boldsymbol{\lambda}_{i,j}$ and $\boldsymbol{\lambda}_{j,i}$ for $e_{i,j} \in E$ in (3.4) and (3.11) to unify the algorithm description. More specifically, we set

$\boldsymbol{\lambda}_{i,j}^0 = \rho_{i,j}^0(\boldsymbol{x}_i^0 - \boldsymbol{x}_j^0)$ at $t = 0$ such that $\boldsymbol{\lambda}_{i,j}^t = -\boldsymbol{\lambda}_{j,i}^t$ holds for all $i = 1, 2, \cdots, N, j \in \mathcal{N}_i$. In this way, we can unify the update rule of agent i without separating $i > j$ and $i < j$ for $j \in \mathcal{N}_i$, as shown in (3.13).

Remark 1. *The proximal Jacobian ADMM* (3.3)-(3.4) *can be considered as a special case of* (3.9)-(3.11) *by assigning the same and constant weight $\rho_{i,j} = \rho$ to different equality constraints $\boldsymbol{x}_i = \boldsymbol{x}_j$. Different from the ADMM which uses the same ρ (which might be time-varying in, e.g., the two-block optimization problem [48]) for all equality constraints, the new approach uses different and time-varying $\rho_{i,j}$ for different equality constraints $\boldsymbol{x}_i = \boldsymbol{x}_j$. As indicated later, this is key for enabling privacy preservation.*

Remark 2. *We did not use Gauss-Seidel update [118], which requires a predefined global order and hence as indicated in [26], is not amenable to parallelism. Different from [26] which has a constant penalty parameter, we intentionally introduce time-varying penalty matrix to enable privacy preservation. Despite enabling new capabilities in privacy protection (with the assistance of partially homomorphic Paillier encryption), introducing time-varying penalty matrix also reduces convergence rate to $O(1/t)$, in contrast to the $o(1/t)$ rate in [26]. Besides giving new capabilities in privacy and different result in convergence rate, the novel idea of intentional time-varying penalty matrix also leads to difference in theoretical analysis in comparison with [26].*

It is obvious that the new ADMM (3.9)-(3.11) can be implemented in a decentralized manner. The detailed implementation procedure is outlined in Algorithm 1.

Remark 3. *A weighted ADMM which also assigns different weights to different equality constraints is proposed in [64]. However, the weights in [64] are constant while Algorithm 1 allows time-varying weights in each iteration, which, as shown later, is key to enable the integration of partially homomorphic cryptography with decentralized optimization.*

3.2.4 Convergence Analysis

In this subsection, we rigorously prove the convergence of Algorithm 1 under the following standard assumptions [118]:

Assumption 1. *Each private local function $f_i : \mathbb{R}^D \to \mathbb{R}$ is convex and continuously differentiable.*

Assumption 2. *Problem* (3.6) *has an optimal solution, i.e., the Lagrangian function*

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x} \tag{3.15}$$

Algorithm 1

Initial Setup: Each agent i initializes $\boldsymbol{x}_i^0$, $\rho_{i,j}^0$.
Input: $\boldsymbol{x}_i^t$, $\boldsymbol{\lambda}_{i,j}^{t-1}$, $\rho_{i,j}^t$
Output: $\boldsymbol{x}_i^{t+1}$, $\boldsymbol{\lambda}_{i,j}^t$, $\rho_{i,j}^{t+1}$

1. Each agent i sends $\boldsymbol{x}_i^t$, $\rho_{i,j}^t$ to its neighboring agents, and then set $\rho_{i,j}^t = \min\{\rho_{i,j}^t, \rho_{j,i}^t\}$. It is clear that $\rho_{i,j}^t = \rho_{j,i}^t$ holds.

2. Each agent i updates $\boldsymbol{\lambda}_{i,j}^t$ as follows for $j \in \mathcal{N}_i$

$$\boldsymbol{\lambda}_{i,j}^t = \boldsymbol{\lambda}_{i,j}^{t-1} + \rho_{i,j}^t(\boldsymbol{x}_i^t - \boldsymbol{x}_j^t). \tag{3.12}$$

   It is clear that $\boldsymbol{\lambda}_{i,j}^t = -\boldsymbol{\lambda}_{j,i}^t$ holds (note that when $t = 0$, we set $\boldsymbol{\lambda}_{i,j}^0 = \rho_{i,j}^0(\boldsymbol{x}_i^0 - \boldsymbol{x}_j^0)$).

3. All agents update their local vectors in parallel:

$$\boldsymbol{x}_i^{t+1} \in \text{argmin}_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2 + \sum_{j \in \mathcal{N}_i} ((\boldsymbol{\lambda}_{i,j}^t)^T \boldsymbol{x}_i + \frac{\rho_{i,j}^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j^t \parallel^2). \tag{3.13}$$

   Here we added two proximal terms $\frac{\rho_{i,i}^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2$ and $\frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^t \parallel^2$ to accommodate the influence of $\boldsymbol{x}_i^t$. For all $\gamma_i > 0$, $\rho_{i,i}^t$ is set to

$$\rho_{i,i}^t = 1 - \sum_{j \in \mathcal{N}_i, j \neq i} \rho_{i,j}^t. \tag{3.14}$$

4. Each agent i updates $\rho_{i,j}^{t+1}$ for all $j \in \mathcal{N}_i$ and sets $t = t + 1$. The detailed update rule for $\rho_{i,j}$ will be elaborated later in Theorem 1.

has a saddle point $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ *such that*

$$L(\boldsymbol{x}^*, \boldsymbol{\lambda}) \leq L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \leq L(\boldsymbol{x}, \boldsymbol{\lambda}^*)$$

holds for all $\boldsymbol{x} \in \mathbb{R}^{ND}$ *and* $\boldsymbol{\lambda} \in \mathbb{R}^{|E|D}$.

Denote the iterating results in the kth step in Algorithm 1 as follows:

$$\boldsymbol{x}^k = [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, ..., \boldsymbol{x}_N^{kT}]^T \in \mathbb{R}^{ND},$$

$$\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D},$$

$$\boldsymbol{\rho}^k = \text{diag}\{\rho_{i,j}^k I_D\}_{ij, e_{i,j} \in E} \in \mathbb{R}^{|E|D \times |E|D}.$$

Further augment the coefficients γ_i $(i = 1, 2, ..., N)$ in (3.13) into the matrix form

$$Q_P = \mathrm{diag}\{\gamma_1, \gamma_2, \ldots, \gamma_N\} \otimes I_D \in \mathbb{R}^{ND \times ND},$$

and augment $\rho_{i,j}^k$ into the following matrix form

$$Q_C^k = \mathrm{diag}\{\sum_{j \in \mathcal{N}_1} \rho_{1,j}^k, \sum_{j \in \mathcal{N}_2} \rho_{2,j}^k, \ldots, \sum_{j \in \mathcal{N}_N} \rho_{N,j}^k\} \otimes I_D,$$

and $Q_C^k \in \mathbb{R}^{ND \times ND}$. By plugging (3.14) into Q_C^k, we have $Q_C^k = I_{ND}$, i.e., Q_C^k is an identity matrix.

Now we are in position to give the main results of this subsection:

Theorem 1. *Under Assumption 1 and Assumption 2, Algorithm 1 is guaranteed to converge to an optimal solution to* (3.6) *if the following two conditions are met:*

Condition A: The sequence $\{\boldsymbol{\rho}^k\}$ satisfies

$$0 \prec \boldsymbol{\rho}^0 \preceq \boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1} \preceq \bar{\boldsymbol{\rho}}, \quad \forall k \geq 0,$$

where $\boldsymbol{\rho}^0 \succ 0$ means that $\boldsymbol{\rho}^0$ is positive definite, and similarly $\boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1}$ means that $\boldsymbol{\rho}^{k+1} - \boldsymbol{\rho}^k$ is positive semi-definite.

Condition B: $Q_P + Q_C^k \succ A^T \bar{\boldsymbol{\rho}} A$.

Proof: The proof is provided in the Appendix A.1. ■

Theorem 2. *The convergence rate of Algorithm 1 is $O(1/t)$, where t is the iteration time.*

Proof: The proof is provided in the Appendix A.2. ■

3.3 Privacy-Preserving Decentralized Optimization

Algorithm 1 requires agents to exchange and disclose states explicitly in each iteration among neighboring agents to reach consensus on the final optimal solution. In this section, we combine partially homomorphic cryptography with Algorithm 1 to propose a privacy-preserving approach for decentralized optimization. We first give the definition of privacy used in this chapter.

Definition 1. *A mechanism $\mathcal{M} : \mathcal{M}(\mathcal{X}) \to \mathcal{Y}$ is defined to be privacy preserving if the input $\mathcal{X}$ cannot be uniquely derived from the output $\mathcal{Y}$.*

This definition of privacy is inspired by the privacy-preservation definitions in [19,27,41,68,71,123] which take advantages of the fact that if a system of equations has infinite number of solutions, it is impossible to derive the exact value of the original input data from the output data. Therefore, privacy preservation is achieved (see, e.g, Part 4.2.2 in [41]). Next, we introduce our privacy-preserving approach based on the Paillier cryptosystem in Sec. 2.2. We combine Paillier cryptosystem with Algorithm 1 to enable privacy preservation in the decentralized solving of optimization problem (3.1). First, note that solving (3.13) amounts to solving the following problem:

$$\nabla f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{i,j}^t + \rho_{i,j}^t(\boldsymbol{x}_i - \boldsymbol{x}_j^t)) + \gamma_i(\boldsymbol{x}_i - \boldsymbol{x}_i^t) = \mathbf{0}. \tag{3.16}$$

Let $\boldsymbol{\lambda}_i = \sum\limits_{j\in N_i} \boldsymbol{\lambda}_{i,j}$, then (3.16) reduces to the following equation

$$\nabla f_i(\boldsymbol{x}_i) + (\sum_{j\in\mathcal{N}_i} \rho_{i,j}^t + \gamma_i)\boldsymbol{x}_i + \boldsymbol{\lambda}_i^t - \sum_{j\in\mathcal{N}_i} \rho_{i,j}^t \boldsymbol{x}_j^t - \gamma_i \boldsymbol{x}_i^t = \mathbf{0}. \tag{3.17}$$

Given that we have set $\rho_{i,i}^t = 1 - \sum\limits_{j\in\mathcal{N}_i, j\neq i} \rho_{i,j}^t$ in (3.14), we can further reduce (3.17) to

$$\nabla f_i(\boldsymbol{x}_i) + (1+\gamma_i)\boldsymbol{x}_i + \boldsymbol{\lambda}_i^t - \sum_{j\in\mathcal{N}_i} \rho_{i,j}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) - (1+\gamma_i)\boldsymbol{x}_i^t = \mathbf{0}. \tag{3.18}$$

By constructing $\rho_{i,j}^t, i \neq j$ as the product of two random positive numbers, i.e., $\rho_{i,j}^t = b_{i\to j}^t \times b_{j\to i}^t = \rho_{j,i}^t$, with $b_{i\to j}^t$ only known to agent i and $b_{j\to i}^t$ only known to agent j, we can propose a privacy-preserving solution to (3.1) based on Algorithm 1, which is described in Algorithm 2.

Several remarks are in order:

1. The only situation that a neighbor knows the state of agent i is when $\boldsymbol{x}_i^t = \boldsymbol{x}_j^t$ is true for $j \in \mathcal{N}_i$. Otherwise, agent i's state $\boldsymbol{x}_i^t$ is encrypted and will not be revealed to its neighbors.

2. Agent i's state $\boldsymbol{x}_i^t$ and its intermediate communication data $b_{j\to i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$ will not be revealed to outside eavesdroppers, since they are encrypted.

3. The state of agent $j \in \mathcal{N}_i$ will not be revealed to agent i, because the decrypted message obtained by agent i is $b_{j\to i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$ with $b_{j\to i}^t$ only known to agent j and varying in each iteration.

4. We encrypt $\mathcal{E}_i(-\boldsymbol{x}_i^t)$ because it is much easier to compute addition in ciphertext. The issue regarding encryption of signed values using Paillier will be addressed in Sec. 3.5.

```
Algorithm 2
Initial Setup: Each agent initializes $\boldsymbol{x}_i^0$.
Input: $\boldsymbol{x}_i^t$, $\boldsymbol{\lambda}_{i,j}^{t-1}$
Output: $\boldsymbol{x}_i^{t+1}$, $\boldsymbol{\lambda}_{i,j}^t$
  1. Agent $i$ encrypts $-\boldsymbol{x}_i^t$ with its public key $k_{pi}$:
         $\boldsymbol{x}_i^t \to \mathcal{E}_i(-\boldsymbol{x}_i^t)$.
     Here the subscript $i$ denotes encryption using the public key of agent $i$.
  2. Agent $i$ sends $\mathcal{E}_i(-\boldsymbol{x}_i^t)$ and its public key $k_{pi}$ to neighboring agents.
  3. Agent $j \in \mathcal{N}_i$ encrypts $\boldsymbol{x}_j^t$ with agent $i$'s public key $k_{pi}$:
         $\boldsymbol{x}_j^t \to \mathcal{E}_i(\boldsymbol{x}_j^t)$.
  4. Agent $j \in \mathcal{N}_i$ computes the difference directly in ciphertext:
         $\mathcal{E}_i(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) = \mathcal{E}_i(\boldsymbol{x}_j^t) \cdot \mathcal{E}_i(-\boldsymbol{x}_i^t)$.
  5. Agent $j \in \mathcal{N}_i$ computes the $b_{j\to i}^t$-weighted difference in ciphertext:
         $\mathcal{E}_i(b_{j\to i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)) = (\mathcal{E}_i(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t))^{b_{j\to i}^t}$.
  6. Agent $j \in \mathcal{N}_i$ sends $\mathcal{E}_i(b_{j\to i}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t))$ back to agent $i$.
  7. Agent $i$ decrypts the message received from $j$ with its private key $k_{si}$ and multiples the result with $b_{i\to j}^t$
     to get $\rho_{i,j}^t(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t)$.
  8. Computing (3.12), agent $i$ obtains $\boldsymbol{\lambda}_{i,j}^t$.
  9. Computing (3.18), agent $i$ obtains $\boldsymbol{x}_i^{t+1}$.
 10. Each agent updates $b_{i\to j}^t$ to $b_{i\to j}^{t+1}$ and sets $t = t + 1$.
```

5. Paillier encryption cannot be performed on vectors directly. For vector messages $\boldsymbol{x}_i^t \in \mathbb{R}^D$, each element of the vector (a real number) has to be encrypted separately. For notation convenience, we still denote it in the same way as scalars, e.g., $\mathcal{E}_i(-\boldsymbol{x}_i^t)$.

6. Paillier cryptosystem only works for integers, so additional steps have to be taken to convert real values in optimization to integers. This may lead to quantization errors. A common workaround is to scale the real value before quantization, as discussed in detail in Sec. 3.5.

7. By incorporating Paillier cryptosystem, it is obvious that the computation complexity and communication load will increase. However, we argue that the privacy provided matters more than this disadvantage

when privacy is of primary concern. Furthermore, our experimental results on Raspberry Pi boards confirm that the added communication and computation overhead is fully manageable on embedded microcontrollers (cf. Sec. 3.7).

8. Our approach is more suitable for small and medium sized optimization problems such as the power system monitoring problem [117] addressed in our prior work.

The key to achieve privacy preservation is to construct $\rho_{i,j}^t, i \neq j$ as the product of two random positive numbers $b_{i\to j}^t$ and $b_{j\to i}^t$, with $b_{i\to j}^t$ generated by and only known to agent i and $b_{j\to i}^t$ generated by and only known to agent j. Next we show that the privacy preservation mechanism does not affect the convergence to the optimal solution.

Theorem 3. *The privacy-preserving algorithm 2 will generate a solution in an ε ball around the optimum if $b_{i\to j}^t$, $b_{j\to i}^t$, and γ_i are updated in the following way (where ε depends on the quantization error):*

1. *$b_{i\to j}^t$ is randomly chosen from $[b_{i\to j}^{t-1}, \bar{b}_{i\to j}]$, with $\bar{b}_{i\to j} > 0$ denoting a predetermined constant only known to agent i;*

2. *γ_i is chosen randomly in the interval $[N\bar{b}^2, \bar{\bar{b}}]$, with $\bar{b} > \max\{\bar{b}_{i\to j}\}$ denoting a predetermined positive constant known to everyone and $\bar{\bar{b}}$ a threshold chosen arbitrarily by agent i and only known to agent i.*

Proof: It can be easily obtained that if $b_{i\to j}^t$ is updated following 1, and γ_i is updated following 2, then Condition A and Condition B in Theorem 1 will be met automatically. Therefore, the states in algorithm 2 should converge to the optimal solution. However, since Paillier cryptosystem only works on unsigned integers, it requires converting real-valued states to integers using e.g., fixed-point arithmetic encoding [2] (after scaled by a large number $N_{\max}$, cf. Sec. 3.5), which leads to quantization errors. The quantization errors lead to numerical errors on the final solution and hence the "ε-ball" statement in Theorem 3. It is worth noting that the numerical error here is no different from the conventional quantization errors met by all algorithms when implemented in practice on a computer. A quantized analysis of the ε-ball is usually notoriously involved and hence we refer interested readers to [136] which is dedicated to this problem. Furthermore, we would like to emphasize that this quantization error can be made arbitrarily small by using an arbitrarily large $N_{\max}$. In fact, our simulation results in Sec. 3.6.2 showed that under $N_{\max} = 10^6$, the final error was on the order of 10^{-14}. ■

3.4 Privacy Analysis

As indicated in the introduction, our approach aims to protect the privacy of agents' intermediate states $\boldsymbol{x}_i^t$s and gradients of f_is as well as the objective functions. In this section, we rigorously prove that these private information cannot be inferred by honest-but-curious adversaries and external eavesdroppers, which are commonly used attack models in privacy studies [62] (cf. definition in Sec. 3.1). It is worth noting that the form of each agent's local objective function can also be totally blind to others, e.g., whether it is a quadratic, exponential, or other forms of convex functions is only known to an agent itself.

As indicated in Sec. 3.3, our approach in Algorithm 2 guarantees that state information is not leaked to any neighbor in one iteration. However, would some information get leaked over time? More specifically, if an honest-but-curious adversary observes carefully its communications with neighbors over several steps, can it put together all the received information to infer its neighbor's state?

We can rigorously prove that an honest-but-curious adversary cannot infer the exact states of its neighbors even by collecting samples from multiple steps.

Theorem 4. *Assume that all agents follow Algorithm 2. Then agent j's exact state value $\boldsymbol{x}_j^k$ cannot be inferred by an honest-but-curious agent i unless $\boldsymbol{x}_i^k = \boldsymbol{x}_j^k$ is true.*

Proof: Suppose that an honest-but-curious agent i collects information from K iterations to infer the information of a neighboring agent j. From the perspective of adversary agent i, the measurements (corresponding to neighboring agent j) seen in each iteration k are $\boldsymbol{y}^k = b_{i\to j}^k b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$ $(k = 0, 1, ..., K)$, i.e., adversary agent i can establish $(K+1)D$ equations based on received information:

$$\begin{cases} \boldsymbol{y}^0 = b_{i\to j}^0 b_{j\to i}^0(\boldsymbol{x}_j^0 - \boldsymbol{x}_i^0), \\ \boldsymbol{y}^1 = b_{i\to j}^1 b_{j\to i}^1(\boldsymbol{x}_j^1 - \boldsymbol{x}_i^1), \\ \qquad \vdots \\ \boldsymbol{y}^{K-1} = b_{i\to j}^{K-1} b_{j\to i}^{K-1}(\boldsymbol{x}_j^{K-1} - \boldsymbol{x}_i^{K-1}), \\ \boldsymbol{y}^K = b_{i\to j}^K b_{j\to i}^K(\boldsymbol{x}_j^K - \boldsymbol{x}_i^K). \end{cases} \tag{3.19}$$

To the adversary agent i, in the system of equations (3.19), $\boldsymbol{y}^k, b_{i\to j}^k, \boldsymbol{x}_i^k$ $(k = 0, 1, 2, ..., K)$ are known, but $\boldsymbol{x}_j^k, b_{j\to i}^k$ $(k = 0, 1, 2, ..., K)$ are unknown. So the above system of $(K+1)D$ equations contains $(K+1)D + K + 1$ unknown variables. It is clear that adversary agent i cannot solve the system of equations (3.19) to infer the exact values of unknowns $\boldsymbol{x}_j^k$ and $b_{j\to i}^k$ $(k = 0, 1, 2, ..., K)$ of agent j. It is worth noting that if for some time index k, $\boldsymbol{x}_j^k = \boldsymbol{x}_i^k$ happens to be true, then adversary agent i will be able to know that agent j has the same state at this time index based on the fact that $\boldsymbol{y}^k$ is $\mathbf{0}$. ■

Using a similar way of reasoning, we can obtain that an honest-but-curious adversary agent i cannot infer the exact gradient of objective function f_j from a neighboring agent j at any point when agent j has another legitimate neighbor other than the honest-but curious neighbor i.

Theorem 5. *In Algorithm 2, the exact gradient of f_j at any point cannot be inferred by an honest-but-curious agent i if agent j has another legitimate neighbor.*

Proof: Suppose that an honest-but-curious adversary agent i collects information from K iterations to infer the gradient of function f_j of a neighboring agent j. The adversary agent i can establish KD equations corresponding to the gradient of f_j by making use of the fact that the update rule (3.18) is publicly known, i.e.,

$$\begin{cases} \nabla f_j(\boldsymbol{x}_j^1) + (1+\gamma_j)\boldsymbol{x}_j^1 + \boldsymbol{\lambda}_j^0 - \sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^0(\boldsymbol{x}_m^0 - \boldsymbol{x}_j^0) - (1+\gamma_j)\boldsymbol{x}_j^0 = \mathbf{0}, \\ \nabla f_j(\boldsymbol{x}_j^2) + (1+\gamma_j)\boldsymbol{x}_j^2 + \boldsymbol{\lambda}_j^1 - \sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^1(\boldsymbol{x}_m^1 - \boldsymbol{x}_j^1) - (1+\gamma_j)\boldsymbol{x}_j^1 = \mathbf{0}, \\ \qquad\qquad \vdots \\ \nabla f_j(\boldsymbol{x}_j^{K-1}) + (1+\gamma_j)\boldsymbol{x}_j^{K-1} + \boldsymbol{\lambda}_j^{K-2} - \sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^{K-2}(\boldsymbol{x}_m^{K-2} - \boldsymbol{x}_j^{K-2}) - (1+\gamma_j)\boldsymbol{x}_j^{K-2} = \mathbf{0}, \\ \nabla f_j(\boldsymbol{x}_j^K) + (1+\gamma_j)\boldsymbol{x}_j^K + \boldsymbol{\lambda}_j^{K-1} - \sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^{K-1}(\boldsymbol{x}_m^{K-1} - \boldsymbol{x}_j^{K-1}) - (1+\gamma_j)\boldsymbol{x}_j^{K-1} = \mathbf{0}. \end{cases} \tag{3.20}$$

In the system of KD equations (3.20), $\nabla f_j(\boldsymbol{x}_j^k)$ $(k = 1, 2, ..., K)$, γ_j, and $\boldsymbol{x}_j^k$ $(k = 0, 1, 2, ..., K)$ are unknown to adversary agent i. Parameters $\boldsymbol{\lambda}_j^k$ and $\sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^k(\boldsymbol{x}_m^k - \boldsymbol{x}_j^k)$ $(k = 0, 1, 2, ..., K-1)$ are known to adversary agent i only when agent j has agent i as the only neighbor. Otherwise, $\boldsymbol{\lambda}_j^k$ and $\sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^k(\boldsymbol{x}_m^k - \boldsymbol{x}_j^k)$ $(k = 0, 1, 2, ..., K-1)$ are unknown to adversary agent i. Noting that $\boldsymbol{\lambda}_j^{k+1} = \boldsymbol{\lambda}_j^k - \sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^{k+1}(\boldsymbol{x}_m^{k+1} - \boldsymbol{x}_j^{k+1})$ and $\boldsymbol{\lambda}_j^0 = -\sum\limits_{m\in\mathcal{N}_j} \rho_{j,m}^0(\boldsymbol{x}_m^0 - \boldsymbol{x}_j^0)$, we can see that the above system of KD equations contains $3KD + D + 1$ unknowns when agent j has more than one neighbor. Therefore, adversary agent i cannot infer the exact values of $\nabla f_j(\boldsymbol{x}_j^k)$ by solving (3.20).

It is worth noting that after the optimization converges, adversary agent i can have another piece of information according to the KKT conditions [26]:

$$\nabla f_j(\boldsymbol{x}_j^*) = -\boldsymbol{\lambda}_j^* \tag{3.21}$$

where $\boldsymbol{x}_j^*$ denotes the optimal solution and $\boldsymbol{\lambda}_j^*$ denotes the optimal multiplier. However, since $\boldsymbol{\lambda}_j^*$ is known to adversary agent i only when agent j has agent i as the only neighbor, we have that adversary agent i cannot infer the exact value of f_j at any point when agent j has another legitimate neighbor besides an honest-but curious neighbor i. ■

Using a similar way of reasoning, we have the following corollary corresponding to the situation where agent j has honest-but-curious agent i as the only neighbor.

Corollary 1. *In Algorithm 2, the exact gradient of f_j at the optimal solution can be inferred by an honest-but-curious agent i if agent j has adversary agent i as the only neighbor. However, at any other point, the gradient of f_j is uninferrable by the adversary agent i.*

Proof: Following a similar line of reasoning of Theorem 5, we can obtain the above Corollary. ■

Based on Theorem 4, Theorem 5, and Corollary 1, we can obtain that agent i cannot infer agent j's local objective function f_j.

Corollary 2. *In Algorithm 2, agent j's local objective function f_j cannot be inferred by an honest-but-curious agent i.*

Proof: According to Theorem 4, Theorem 5, and Corollary 1, the intermediate states and corresponding gradients of f_j cannot be inferred by adversary i. Therefore, adversary i cannot infer agent j's local objective function f_j as well. ■

Furthermore, we have that an external eavesdropper cannot infer any private information of all agents.

Corollary 3. *All agents' intermediate states, gradients of objective functions, and objective functions cannot be inferred by an external eavesdropper.*

Proof: Since all exchanged messages are encrypted and that cracking the encryption is practically infeasible [36], an external eavesdropper cannot learn anything by intercepting these messages. Therefore, it cannot infer any agent's intermediate states, gradients of objective functions, and objective functions. ■

From the above analysis, it is obvious that agent j's private information cannot be uniquely derived by adversaries. However, an honest-but-curious neighbor i can still get some range information about the state $\boldsymbol{x}_j^k$ and this range information will become tighter as $\boldsymbol{x}_j^k$ converges to the optimal solution as $k \to \infty$ (cf. the simulation results in Fig. 3.4). We argue that this is completely unavoidable for any privacy-preserving approaches because all agents have to agree on the same final state, upon which the privacy of $\boldsymbol{x}_j^k$ will disappear. In fact, this is also acknowledged in [54], which shows that the privacy of $\boldsymbol{x}_j^k$ will vanish as $k \to \infty$ and the noise variance converges to zero at the state corresponding to the optimal solution. It is worth noting that when the constraint is of a form different from consensus, it may be possible to protect the privacy of $\boldsymbol{x}_j^k$ when $k \to \infty$. However, how to incorporate the proposed privacy mechanism in decentralized optimization under non-consensus constraint is difficult and could be addressed in future work.

Remark 4. *It is worth noting that an adversary agent i can combine systems of equations* (3.19) *and* (3.20) *to infer the information of a neighboring agent j. However, this will not increase the ability of adversary agent i because the combination will not change the fact that the number of unknowns is greater than the number of establishable relevant equations. In addition, if all other agents collude to infer $\boldsymbol{x}_j^k$ of agent j, these agents can be considered as one agent which amounts to having a network consisting of two agents.*

Remark 5. *From Theorem 4, we can see that in decentralized optimization, an agent's information will not be disclosed to other agents no matter how many neighbors it has. This is in distinct difference from the average consensus problem in [80, 98] where privacy cannot be protected for an agent if it has an honest-but-curious adversary as the only neighbor.*

3.5 Implementation Details

In this section, we discuss several technical issues that have to be addressed in the implementation of Algorithm 2.

1. In modern communication, a real number is represented by a floating point number, while encryption techniques only work for unsigned integers. To deal with this problem, we uniformly multiplied each element of the vector message $\boldsymbol{x}_i^t \in \mathbb{R}^D$ (in floating point representation) by a sufficiently large number $N_{\max}$ and round off the fractional part during the encryption to convert it to an integer. After decryption, the result is divided by $N_{\max}$. This process is conducted in each iteration and this quantization brings an error upper-bounded by $\frac{1}{N_{\max}}$. In implementation, $N_{\max}$ can be chosen according to the used data structure.

2. As indicated in 1, encryption techniques only work for unsigned integers. In our implementation all

integer values are stored in fix-length integers (i.e., long int in C) and negative values are left in 2's complement format. Encryption and intermediate computations are carried out as if the underlying data were unsigned. When the final message is decrypted, the overflown bits (bits outside the fixed length) are discarded and the remaining binary number is treated as a signed integer which is later converted back to a real value.

3.6 Numerical Simulations

In this section, we first illustrate the efficiency of the proposed approach using C/C++ implementations. Then we compare our approach with the algorithm in [54] and the algorithm in [71]. The open-source C implementation of the Paillier cryptosystem [12] is used in our simulations. We conducted numerical experiments on the following global objective function

$$f(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{N} \frac{1}{p_i} \parallel H_i \tilde{\boldsymbol{x}} - \boldsymbol{\theta}_i \parallel^2, \tag{3.22}$$

which makes the optimization problem (3.1) become

$$\min_{\tilde{\boldsymbol{x}}} \qquad \sum_{i=1}^{N} \frac{1}{p_i} \parallel H_i \tilde{\boldsymbol{x}} - \boldsymbol{\theta}_i \parallel^2 \tag{3.23}$$

with $\boldsymbol{\theta}_i \in \mathbb{R}^D$, $H_i = h_i \boldsymbol{I}_D$ ($h_i \in \mathbb{R}$), and $p_i > 0$ ($p_i \in \mathbb{R}$). Hence, each agent i deals with a private local objective function

$$f_i(\boldsymbol{x}_i) = \frac{1}{p_i} \parallel H_i \boldsymbol{x}_i - \boldsymbol{\theta}_i \parallel^2, \forall i \in \{1, 2, \ldots, N\}. \tag{3.24}$$

We used the above function (3.22) because it is easy to verify whether the obtained solution is the minimal value of the original optimization problem, which should be $\frac{\sum_{i=1}^{N} \frac{2h_i}{p_i} \boldsymbol{\theta}_i}{\sum_{i=1}^{N} \frac{2h_i^2}{p_i}}$. Furthermore, (3.22) makes it easy to compare with [54], whose verification is also based on (3.22).

In the implementation, the parameters are set as follows: $N_{\max}$ was set to 10^6 to convert each element in $\boldsymbol{x}_i$ to a 64-bit integer during intermediate computations. $b_{i \to j}^t$ was also scaled up in the same way and represented by a 64-bit integer. The encryption and decryption keys were chosen as 256-bit long.

3.6.1 Evaluation of Our Approach

We implemented Algorithm 2 on different network topologies, all of which gave the right optimal solution. Simulation results confirmed that our approach always converged to the optimal solution of (3.23). Fig. 3.2 visualizes the evolution of $\boldsymbol{x}_i$ $(i = 1, 2, ..., 6)$ in one specific run where the network deployment is illustrated in Fig. 3.1. In Fig. 3.2, x_{ij} $(i = 1, 2, ..., 6, j = 1, 2)$ denotes the jth element of $\boldsymbol{x}_i$. All $\boldsymbol{x}_i$ $(i = 1, 2, ..., 6)$ converged to the optimal solution $[38.5; \frac{407}{6}]$. In this run, $\bar{b}$ was set to 0.65 and γ_is were set to 3. Fig. 3.3 visualizes the encrypted weighted differences (in ciphertext) $\mathcal{E}_1(b^t_{2\to1}(\boldsymbol{x}^t_{21} - \boldsymbol{x}^t_{11}))$,

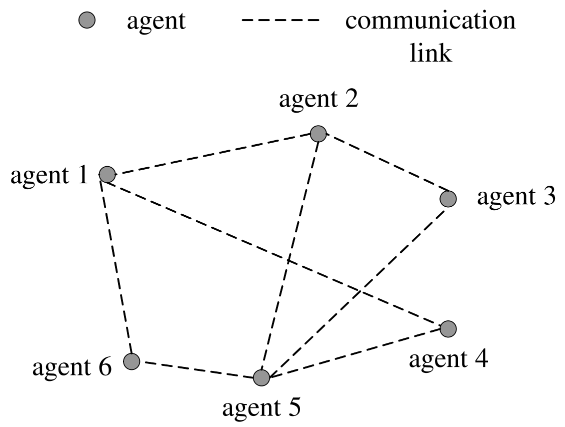


Figure 3.1: A network of six agents ($N = 6$).

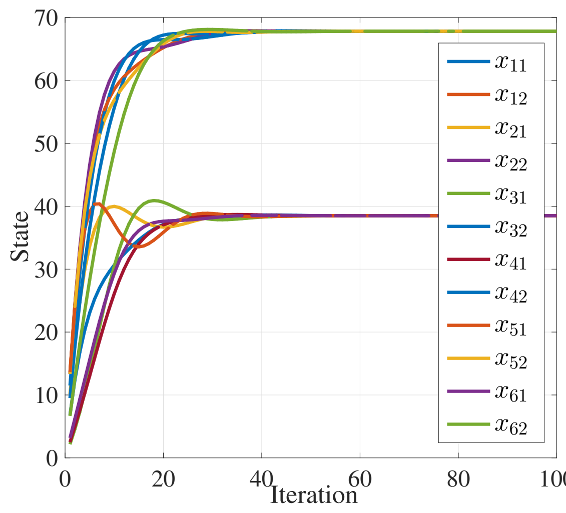


Figure 3.2: The evolution of $\boldsymbol{x}_i$ $(i = 1, 2, ..., 6)$ in Algorithm 2.

$\mathcal{E}_1(b^t_{4\to1}(\boldsymbol{x}^t_{41} - \boldsymbol{x}^t_{11}))$, and $\mathcal{E}_1(b^t_{6\to1}(\boldsymbol{x}^t_{61} - \boldsymbol{x}^t_{11}))$. It is worth noting that although the states of all agents have converged after about 40 iterations, the encrypted weighted differences (in ciphertext) still appeared random to an outside eavesdropper.

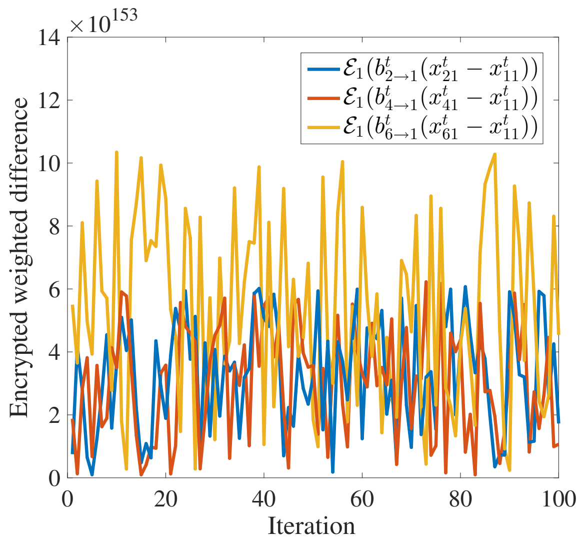


Figure 3.3: The evolution of the encrypted weighted differences (in ciphertext) $\mathcal{E}_1(b^t_{2\to1}(\boldsymbol{x}^t_{21} - \boldsymbol{x}^t_{11}))$, $\mathcal{E}_1(b^t_{4\to1}(\boldsymbol{x}^t_{41} - \boldsymbol{x}^t_{11}))$, and $\mathcal{E}_1(b^t_{6\to1}(\boldsymbol{x}^t_{61} - \boldsymbol{x}^t_{11}))$ in Algorithm 2.

We also simulated an honest-but-curious adversary who tries to estimate its neighbors' intermediate states and gradients in order to estimate the objective function. We considered the worse case of two agents (A and B) where agent B is the honest-but-curious adversary and intends to estimate the objective function f_A of agent A. The individual local objective functions are the same as (3.24) with $\theta_i \in \mathbb{R}$. Because agent B knows the constraints on agent A's generation of $b^t_{A\to B}$ and γ_A (cf. Theorem 3), it generates estimates of $b^t_{A\to B}$ and γ_A in the same random way. Then it obtained a series of estimated x^t_A and $\triangledown f_A(x^t_A)$ according to (3.20). Finally, agent B used the estimated x^t_A and $\triangledown f_A(x^t_A)$ to estimate f_A.

Fig. 3.4 and Fig. 3.5 show the estimated x_A and f_A in 2,000 trials when agent B used simple linear regression to estimate $\triangledown f_A(x)$. Fig. 3.5 suggests that agent B cannot get a good estimate of f_A. Moreover, it is worth noting that all these estimated functions give the same optimal solution as f_A to the optimization problem (3.23).

In addition, the encryption/decryption computation took about 1ms for each agent to communicate with one neighbor at each iteration on a 3.6 GHz CPU, which is manageable in small or medium sized real-time optimization problems such as the power system monitoring problem [117] addressed in our prior work. For

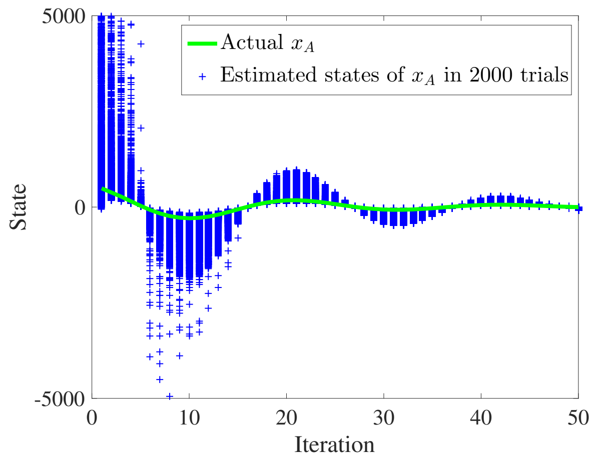


Figure 3.4: Estimated states of x_A in 2,000 trials.

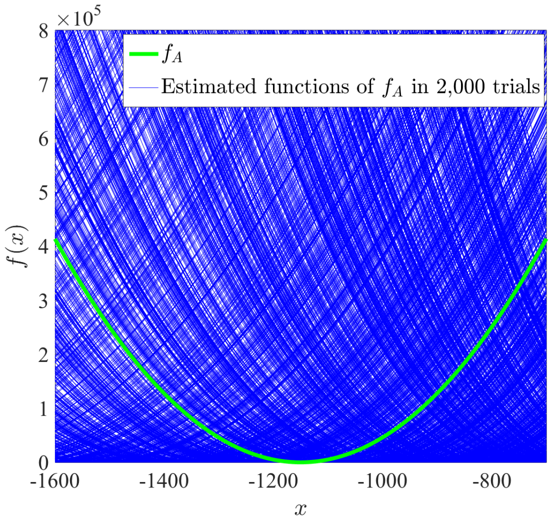


Figure 3.5: Estimated functions of f_A in 2,000 trials.

large sized optimization problems like machine learning with extremely large dimensions, the approach may be computationally too heavy due to the underlying Paillier encryption scheme.

3.6.2 Comparison with the algorithm in [54]

We then compared our approach with the differential-privacy based privacy-preserving optimization algorithm in [54]. Under the communication topology in Fig. 3.1, we simulated the algorithm in [54] under seven different privacy levels: $\epsilon = 0.2, 1, 10, 20, 30, 50, 100$. The global function we used for comparison was (3.22) with p_i $(i = 1, 2, .., 6)$ fixed to 2, h_i $(i = 1, 2, .., 6)$ fixed to 1, and $\boldsymbol{\theta}_i = [0.1 \times (i-1) + 0.1; 0.1 \times (i-1) + 0.2]$. The domain of optimization was set to $\mathcal{X} = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}$ for the algorithm in [54]. Note that the optimal solution $[0.35; 0.45]$ resided in $\mathcal{X}$. Parameter settings for the algorithm in [54] are detailed as follows: $n = 2$, $c = 0.5$, $q = 0.8$, $p = 0.9$, and

$$a_{ij} = \begin{cases} 0.2 & j \in \mathcal{N}_i \backslash i, \\ 0 & j \notin \mathcal{N}_i, \\ 1 - \sum\limits_{j \in \mathcal{N}_i \backslash i} a_{ij} & i = j, \end{cases} \tag{3.25}$$

for $i = 1, 2, ..., 6$. Here $\mathcal{N}_i \backslash i$ denotes all values except i in set $\mathcal{N}_i$. Furthermore, we used the performance index d in [54] to quantify the optimization error, which was computed as the average value of squared distances with respect to the optimal solution over M runs [54], i.e.,

$$d = \frac{\sum\limits_{i=1}^{6} \sum\limits_{k=1}^{M} \| \boldsymbol{x}_i^k - [0.35; 0.45] \|^2}{6M}$$

with $\boldsymbol{x}_i^k$ the obtained solution of agent i in the kth run.

Simulation results from 5,000 runs showed that our approach converged to $[0.35; 0.45]$ with an error $d = 3.14 \times 10^{-14}$, which is negligible compared with the simulation results under the algorithm in [54] (cf. Fig. 3.6, where each differential privacy level was implemented for 5,000 times). The results confirm the trade-off between privacy and accuracy for differential-privacy based approaches and demonstrate the advantages of our approach in terms of optimization accuracy.

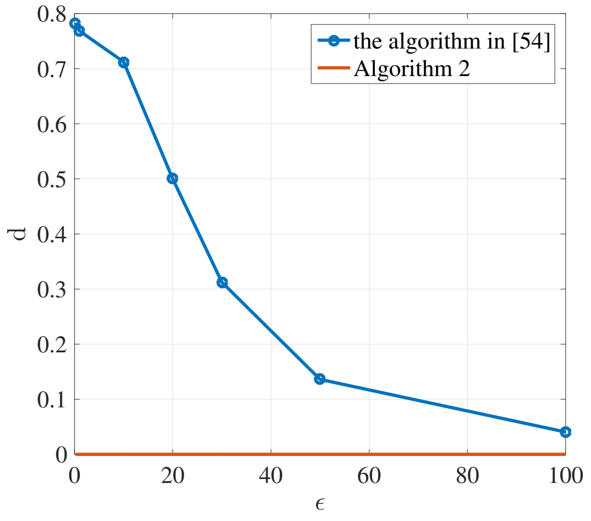


Figure 3.6: The comparison of Algorithm 2 with the algorithm in [54] in terms of optimization error.

3.6.3 Comparison with the algorithm in [71]

We also compared our approach with the privacy-preserving optimization algorithm in [71]. The network communication topology used for comparison is still the one in Fig. 3.1 and the global objective function used is (3.22) with p_i $(i = 1, 2, .., 6)$ fixed to 2, h_i $(i = 1, 2, .., 6)$ fixed to 1, and $\theta_i \in \mathbb{R}^2$. The adjacency matrix of network graph is defined in (3.25) for the algorithm in [71]. Moreover, we let every agent update at each iteration and $c_i = 1$ $(i = 1, ..., 6)$ for [71]. The initial states are set to the same values for both algorithms.

Fig. 3.7 and Fig. 3.8 show the evolution of $\boldsymbol{x}_i$ in our approach and the algorithm in [71] respectively. It is clear that our approach converged faster than the algorithm in [71].

3.7 Implementation on Raspberry PI boards

We also implemented our privacy-preserving approach on twelve Raspberry Pi boards to confirm the efficiency of the approach in real-world physical systems. Each board has 64-bit ARMv8 CPU and 1 GB RAM (cf. Fig. 3.9). The optimization problem (3.23) was used in implementation with p_i $(i = 1, 2, .., 6)$ fixed to 2, h_i $(i = 1, 2, .., 6)$ fixed to 1, and $\theta_i \in \mathbb{R}$. In the implementation, "libpaillier-0.8" library [3] was used to realize the Paillier encryption and decryption process, "sys/socket.h" C library was used to conduct communication through Wi-Fi, and "pthread" C library was used to generate multiple parallel threads to realize parallelism in

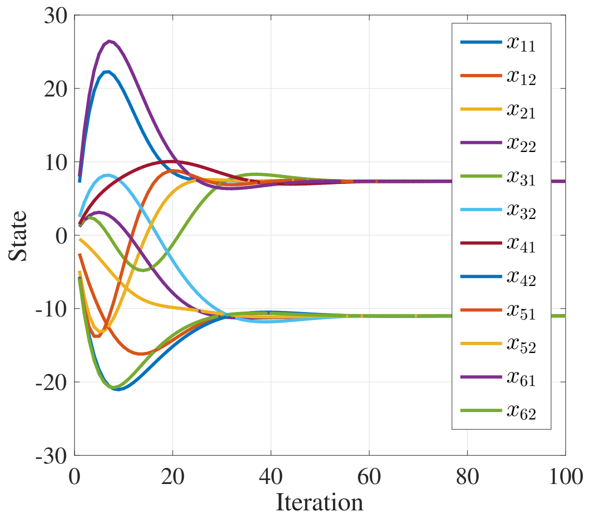


Figure 3.7: The evolution of $\boldsymbol{x}_i$ in Algorithm 2.

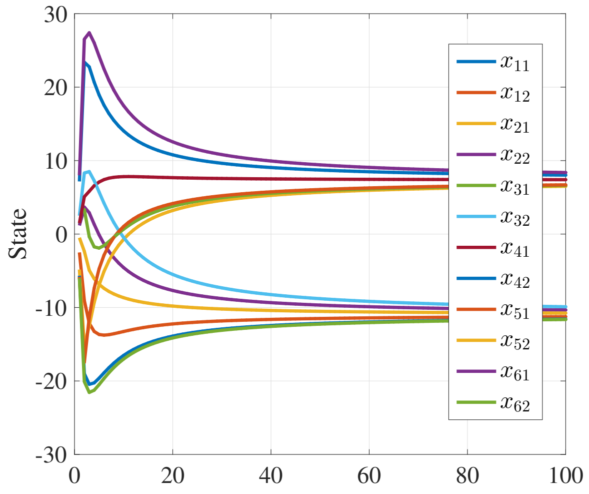


Figure 3.8: The evolution of $\boldsymbol{x}_i$ in the algorithm of [71].

multi-agent networks. The encryption and decryption keys were chosen as 512-bit long.

Implementation results confirmed that our approach always converged to the optimal solution. Fig. 3.10 visualizes the evolution of x_i $(i = 1, 2, ..., 12)$ in one specific implementation where the network topology used is a cycle graph. We can see that each x_i converged to the optimal solution 188.417.

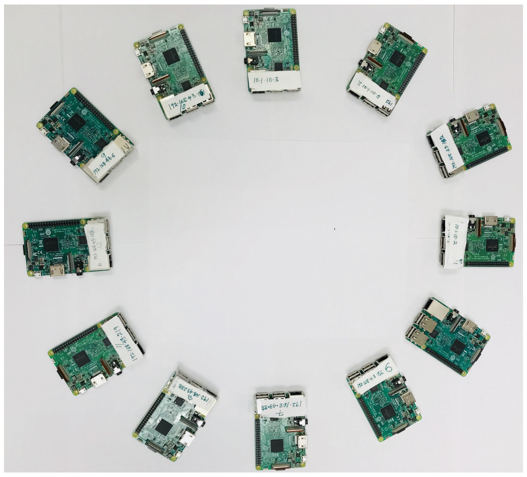


Figure 3.9: The twelve Raspberry Pi boards

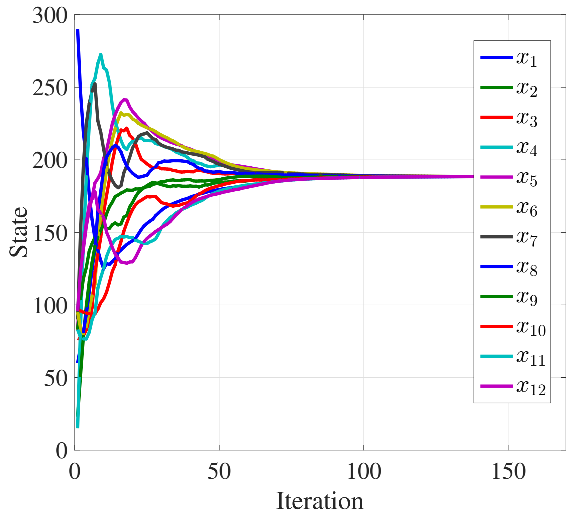


Figure 3.10: The evolution of x_i of Algorithm 2 in the experimental verification using Raspberry Pi boards.

3.8 Summaries

In this chapter, we presented a privacy-preserving decentralized optimization approach by proposing a new ADMM and leveraging partially homomorphic cryptography. By incorporating Paillier cryptosystem into the newly proposed decentralized ADMM, our approach provides guarantee for privacy preservation without compromising the solution in the absence of any aggregator or third party. This is in sharp contrast to differential-privacy based approaches which protect privacy through injecting noise and are subject to a fundamental trade-off between privacy and accuracy. Different from the privacy-preserving optimization approach in [71] which only protects the privacy of gradients, our approach preserves the privacy of both intermediate states and gradients. In addition, [71] assumes that an adversary does not have access to the adjacency matrix of the network graph while our approach does not need this assumption.Theoretical analysis confirms that an honest-but-curious adversary cannot infer the information of neighboring agents even by recording and analyzing the information exchanged in multiple iterations. The new ADMM allows time-varying penalty matrices and have a theoretically guaranteed convergence rate of $O(1/t)$, which makes it of mathematical interest by itself. Numerical and experimental results are given to confirm the effectiveness and efficiency of the proposed approach.

Chapter 4

Privacy-preserving Decentralized Optimization Based on Subgradient Method

4.1 Introduction

This chapter considers a decentralized problem in which N agents cooperatively solve a constrained optimization over a time-varying network topology. Such optimization problem have found applications in domains as diverse as source localization in sensor networks [78], spectrum sensing in cognitive networks [128], support vector machine in machine learning [23], cooperative control [87], data regression in statistics [67, 77], and the monitoring of power systems [35, 83]. In this chapter, we propose an approach that enables privacy-preservation in decentralized optimization through incorporating partially homomorphic cryptography in subgradient method. We show that by employing the convergence property of subgradient method, cryptographic techniques can be incorporated in a fully decentralized manner to enable privacy-preservation in decentralized optimization in the absence of any third party or aggregator.

We also consider the two types of adversaries defined in 3.1 in this chapter, which are *Honest-but-curious adversaries* who follow all protocol steps correctly but are curious and collect all intermediate and input/output data in an attempt to learn some information about other participating agents [36, 62] and *External eavesdroppers* who steal information through wiretapping all communication channels and intercepting exchanged messages between agents. In addition, we define privacy as preserving the confidentiality of agents' intermediate states and objective functions in this chapter.

The rest of this chapter is organized as follows: Sec. 4.2 reviews the constrained decentralized optimization problem. Based on the Paillier cryptosystem, a completely decentralized privacy-preserving approach is proposed in Sec. 4.3. Rigorous analysis of the guaranteed privacy under the approach is addressed in Sec. 4.4 and its implementation details are discussed in Sec. 4.5. Sec. 4.6 discusses its application to the average consensus problem. Numerical simulation results are given in Sec. 4.7 to confirm the effectiveness and computational efficiency of the proposed approach. In the end, we draw summaries in Sec. 4.8.

4.2 Preliminaries

4.2.1 Constrained Decentralized Optimization

4.2.1.1 Optimization Model

The problem of constrained decentralized optimization can be formulated in the following form:

$$\begin{aligned} \min_{\tilde{\boldsymbol{x}}} \qquad & \sum_{i=1}^{N} f_i(\tilde{\boldsymbol{x}}) \\ \text{subject to} \qquad & \tilde{\boldsymbol{x}} \in \mathcal{X}, \end{aligned} \tag{4.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^D$ is a closed convex set common to all agents, and function $f_i : \mathbb{R}^D \to \mathbb{R}$ is the local objective function private to (only known to) agent i. Note that here f_i is convex but not necessarily differentiable everywhere.

Remark 6. *In practical applications, the solutions to an unconstrained optimization problem should be finite. Therefore, we can easily reformulate an unconstrained optimization problem as a constrained optimization problem by setting a large enough constraint set $\mathcal{X}$.*

The constrained optimization problem has been widely used. Here we give some typical examples.

Example 1: In source localization, N sensors cooperatively localize the position of a source using noisy range measurements with respect to the source from distributedly deployed sensors. When the source is located in the convex hull of deployed sensors, a classical approach is to turn the localization problem into the problem of finding a point common to a set of closed convex sets X_i [78], which can be formulated as follows:

$$\min_{\boldsymbol{x}} \qquad \frac{1}{2}\sum_{i=1}^{N} \parallel \boldsymbol{x} - P_{X_i}[\boldsymbol{x}] \parallel^2$$

where $i = 1, 2, ..., N$ is the index of deployed sensors and

$$X_i = \{\boldsymbol{x} \in \mathbb{R}^D | \parallel \boldsymbol{x} - \boldsymbol{p}_i \parallel^2 \leq r_i^2\}.$$

Here, $\boldsymbol{p}_i$ is the position of sensor i and r_i is the range measurement of the source with respect to sensor i. $P_{X_i}[\cdot]$ denotes the projection operation onto the set X_i, i.e., $P_{X_i}[\boldsymbol{r}] = \underset{\boldsymbol{y} \in X_i}{\operatorname{argmin}} \parallel \boldsymbol{y} - \boldsymbol{r} \parallel$.

Example 2: In cooperative control, a typical problem is to guarantee that a group of agents reach a common decision or agreement formulated as follows [54, 87]:

$$\begin{aligned} \min_{\boldsymbol{x}} \qquad & \sum_{i=1}^{N} \frac{1}{2} \parallel \boldsymbol{x} - \boldsymbol{\theta}_i \parallel^2 \\ \text{subject to} \qquad & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{\theta}_i$ $(i = 1, 2, ..., N)$ are known parameters, and $\boldsymbol{x}$ is the unknown vector.

Example 3: In statistical analysis, many problems have the constrained optimization formulation in (4.1):

Simple linear regression

$$\min_{\gamma_0, \gamma_1} \qquad \sum_{i=1}^{N} \parallel y_i - \gamma_0 - \gamma_1 x_i \parallel^2,$$

where y_i and x_i are known parameters, γ_0 and γ_1 are unknown variables.

Logistic regression [67]

$$\min_{\boldsymbol{x}, c} \qquad \frac{1}{N}\sum_{i=1}^{N} \log(1 + \exp(-b_i(\boldsymbol{x}^T \boldsymbol{a}_i + c))) + \tau \parallel \boldsymbol{x} \parallel_1,$$

where b_i, $\boldsymbol{a}_i$, and τ are known parameters, c and $\boldsymbol{x}$ are unknown variables.

Other examples can be found in power systems [83], compressive spectrum sensing [128], and machine leaning [23, 123, 134].

4.2.1.2 Decentralized Algorithm [87]

A decentralized solution to the constrained optimization problem (4.1) is the projected subgradient algorithm in [87]. In the projected subgradient algorithm, each agent i updates its estimate by first fusing the estimates from its neighbors, then taking a subgradient step, and finally projecting to the closed convex set $\mathcal{X}$, i.e.,

$$\boldsymbol{v}_i^k = \sum_{j=1}^{N} a_{ij}^k \boldsymbol{x}_j^k, \tag{4.3}$$

$$\boldsymbol{x}_i^{k+1} = P_{\mathcal{X}}[\boldsymbol{v}_i^k - \alpha_k d_i^k]. \tag{4.4}$$

Here, a_{ij} is a nonnegative weight assigned to $\boldsymbol{x}_j$ by agent i, $P_{\mathcal{X}}[\cdot]$ denotes the projection operation onto the set $\mathcal{X}$, i.e., $P_{\mathcal{X}}[\boldsymbol{r}] = \underset{\boldsymbol{y}\in\mathcal{X}}{\operatorname{argmin}} \parallel \boldsymbol{y} - \boldsymbol{r} \parallel$, $\alpha_k > 0$ is a stepsize, and d_i^k is a subgradient of f_i at $\boldsymbol{x} = \boldsymbol{v}_i^k$.

4.2.1.3 Convergence Analysis

According to [87], the algorithm (4.3)-(4.4) is guaranteed to converge under the following three assumptions when the stepsize α_k satisfies $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$:

Assumption 3. *There exists a scalar $0 < \eta < 1$ such that for all $k \geq 0$ and $i = 1, 2, ..., N$:*

1. $a_{ii}^k \geq \eta$.
2. $a_{ij}^k \geq \eta$ *for all $j \in \mathcal{N}_i^k$. Here $\mathcal{N}_i^k$ denotes the set of all neighboring agents of i at time instant k.*
3. $a_{ij}^k = 0$ *for all $j \notin \mathcal{N}_i^k \cup \{i\}$.*
4. $\sum_{j=1}^{N} a_{ij}^k = 1$.
5. $a_{ij}^k = a_{ji}^k$.

Note that the N agents may form a time-varying network, i.e., the neighbors of agent i may change with time.

Assumption 4. *The graph (V, E_∞) is strongly connected, where*

$$E_\infty = \{e_{ij} | e_{ij} \in E_k \textit{ for infinitely many indices } k\}.$$

Here, E_k denotes the set of communication links (undirected edges) at time instant k, and $e_{ij} \in E_k$ denotes that agents i and j are neighbors (directly connected) at time instant k. Note that here we denote a communication link as e_{ij} if $i < j$ is true or as e_{ji} otherwise.

Assumption 5. *There exists an integer $B \geq 1$ such that for each $e_{ij} \in E_\infty$, agent j sends its estimate to agent i at least once every B consecutive time slots.*

A typical way to choose α_k is $\alpha_k = \frac{T_1}{k+T_2}$, where $0 < T_1 < \infty$ and $0 < T_2 < \infty$.

4.3 Privacy-preserving Decentralized Optimization

In the algorithm (4.3)-(4.4) for the constrained optimization problem (4.1), to reach consensus on the final optimal solution, agents exchange and disclose estimates (states) explicitly in each iteration to neighboring agents, which leads to privacy breaches. Such information exchange is also vulnerable to eavesdropping attacks which aim to steal information by intercepting exchanged messages. In this section, we introduce a completely decentralized and third-party free approach to enable privacy-preservation in the constrained decentralized optimization problem. More specifically, we will propose an interaction protocol which enables an easy integration of partially homomorphic cryptography with the algorithm (4.3)-(4.4) to enable privacy-preservation without the assistance of any aggregator or third party. The definition of privacy is given in Definition 1.

To this end, we first rewrite (4.3) as follows

$$\boldsymbol{v}_i^k = \boldsymbol{x}_i^k + \sum_{j=1, j\neq i}^{N} a_{ij}^k (\boldsymbol{x}_j^k - \boldsymbol{x}_i^k). \tag{4.5}$$

The key idea of our privacy-preserving mechanism is to construct $a_{ij}^k, i \neq j, j \in \mathcal{N}_i^k$ as the product of two random positive numbers, i.e., $a_{ij}^k = b_{i\to j}^k \times b_{j\to i}^k = a_{ji}^k$, with $b_{i\to j}^k$ generated by and only known to agent i, and $b_{j\to i}^k$ generated by and only known to agent j when agent i and agent j can communicate with each other at time instant k. It is worth noting that if agent i and agent j cannot communicate with each other at time instant k, a_{ij}^k and a_{ji}^k are set to 0 directly. Next we give in detail our privacy-preserving solution to the constrained-decentralized-optimization problem in (4.1), which is described in Algorithm 3.

Several remarks are in order:

```
Algorithm 3
Initial Setup: Each agent initializes $\boldsymbol{x}_i^0$.
Input: $\boldsymbol{x}_i^k$
Output: $\boldsymbol{x}_i^{k+1}$

1. Agent $i$ encrypts $-\boldsymbol{x}_i^k$ with its public key $k_{pi}$:
       $\boldsymbol{x}_i^k \rightarrow \mathcal{E}_i(-\boldsymbol{x}_i^k).$
   Here the subscript $i$ denotes encryption using the public key of agent $i$.
2. Agent $i$ sends $\mathcal{E}_i(-\boldsymbol{x}_i^k)$ and its public key $k_{pi}$ to its neighboring agents.
3. Agent $j \in \mathcal{N}_i^k$ encrypts $\boldsymbol{x}_j^k$ with agent $i$'s public key $k_{pi}$:
       $\boldsymbol{x}_j^k \rightarrow \mathcal{E}_i(\boldsymbol{x}_j^k).$
4. Agent $j \in \mathcal{N}_i^k$ computes the difference directly in ciphertext:
       $\mathcal{E}_i(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k) = \mathcal{E}_i(\boldsymbol{x}_j^k) \cdot \mathcal{E}_i(-\boldsymbol{x}_i^k).$
5. Agent $j \in \mathcal{N}_i^k$ computes the $b_{j\to i}^k$-weighted difference in ciphertext:
       $\mathcal{E}_i(b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)) = (\mathcal{E}_i(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k))^{b_{j\to i}^k}.$
6. Agent $j \in \mathcal{N}_i^k$ sends $\mathcal{E}_i(b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k))$ back to agent $i$.
7. Agent $i$ decrypts the message received from $j$ with its private key $k_{si}$ and multiples the result with $b_{i\to j}^k$
   to get $a_{ij}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$.
8. Computing (4.5), agent $i$ obtains $\boldsymbol{v}_i^k$.
9. Computing (4.4), agent $i$ obtains $\boldsymbol{x}_i^{k+1}$.
10. Each agent updates $b_{i\to j}^k$ to $b_{i\to j}^{k+1}$ and sets $k = k + 1$.
```

1. Agent i's state $\boldsymbol{x}_i^k$ and its intermediate communication data $b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$ will not be revealed to outside eavesdroppers, since they are encrypted.

2. The state of agent $j \in \mathcal{N}_i^k$ will not be revealed to agent i, because the decrypted message obtained by agent i is $b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$ with $b_{j\to i}^k$ only known to agent j and varying in each iteration.

3. We encrypt $\mathcal{E}_i(-\boldsymbol{x}_i^k)$ because it is much easier to compute addition in ciphertext. The issue regarding encryption of signed values using Paillier will be addressed in Sec. 4.5.

4. Paillier encryption cannot be performed on vectors directly. For vector messages $\boldsymbol{x}_i^k \in \mathbb{R}^D$, each element of the vector has to be encrypted separately. For notational convenience, we still denote it in the

same way as scalars, e.g., $\mathcal{E}_i(-\boldsymbol{x}_i^k)$.

5. Paillier cryptosystem only works for integers, so additional steps have to be taken to convert real values in optimization to integers. This may lead to quantization errors. A common workaround is to scale a real value before quantization, as discussed in detail in Sec. 4.5.

6. The proposed approach requires agents to update synchronously and it may fail to converge if applied to asynchronous networks directly.

In Algorithm 3, steps 1 to 7 constitute the core of our approach to incorporating Paillier cryptosystem in privacy-preserving optimization in a fully decentralized manner. In fact, it can also be seen that the only net effect of our privacy-preserving mechanism is random and time-varying coefficients a_{ij}^k in (4.5). Next we show that the privacy-preserving mechanism does not affect the convergence of the algorithm to its optimal solution.

Theorem 6. *The convergence of the privacy-preserving Algorithm 3 is guaranteed if Assumptions 4 and 5 hold, all $b_{i\to j}^k$ are randomly chosen from $[\sqrt{\eta}, \sqrt{\frac{1-\eta}{N-1}}]$ with $0 < \eta < 1/N$, and the stepsize α_k satisfies $\sum\limits_k \alpha_k = \infty$ and $\sum\limits_k \alpha_k^2 < \infty$.*

Proof: We show that Assumption 3 will be met if all $b_{i\to j}^k$ are randomly chosen from $[\sqrt{\eta}, \sqrt{\frac{1-\eta}{N-1}}]$ with $0 < \eta < 1/N$. First, since a_{ij}^k and a_{ji}^k are set to 0 directly when $j \notin \mathcal{N}_i^k$, and to $a_{ij}^k = b_{i\to j}^k b_{j\to i}^k = b_{j\to i}^k b_{i\to j}^k = a_{ji}^k$ when $j \in \mathcal{N}_i^k$, it is clear that conditions 3) and 5) in Assumption 3 are satisfied. (Note that $b_{i\to j}^k$ and $b_{j\to i}^k$ are unknown to agents j and i respectively, so a_{ij}^k and a_{ji}^k are equal but unknown to both agent i and agent j.) Next, rewriting (4.5) as

$$\boldsymbol{v}_i^k = (1 - \sum_{j=1, j\neq i}^{N} a_{ij}^k)\boldsymbol{x}_i^k + \sum_{j=1, j\neq i}^{N} a_{ij}^k \boldsymbol{x}_j^k$$

we have that $a_{ii}^k = 1 - \sum\limits_{j=1,j\neq i}^{N} a_{ij}^k$ is always true under the update rule and hence the condition 4) of Assumption 3 is satisfied. When $j \in \mathcal{N}_i^k$, $b_{i\to j}^k$ and $b_{j\to i}^k$ are chosen from the interval $[\sqrt{\eta}, \sqrt{\frac{1-\eta}{N-1}}]$, so we have $\eta \le a_{ij}^k = b_{i\to j}^k b_{j\to i}^k \le \frac{1-\eta}{N-1}$ and further $a_{ii}^k = 1 - \sum\limits_{j=1,j\neq i}^{N} a_{ij}^k \ge 1 - (N-1) \times \sqrt{\frac{1-\eta}{N-1}} \times \sqrt{\frac{1-\eta}{N-1}} = \eta$, i.e., $a_{ij}^k \ge \eta$ and $a_{ii}^k \ge \eta$ for $0 < \eta < 1/N$. Therefore, the conditions 1) and 2) in Assumption 3 are also satisfied. So we have Theorem 6. ■

Remark 7. *It is worth noting that in Algorithm 3, if the state of agent i and agent j, i.e., $\boldsymbol{x}_i^k$ and $\boldsymbol{x}_j^k$ happen to be equal to each other, then agent i will be able to know this based on the fact that the obtained $a_{ij}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$*

in step 7 is zero. This fact that agent i and agent j being equal is inferable from zero $a_{ij}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$ can be covered by allowing $b_{i\to j}^k$ and $b_{j\to i}^k$ to be set to zero randomly. In this case, agents i and j's mutual link is intentionally abandoned and they are not neighbors any more at this specific time instant. Because they are not neighbors at this time instant, having $a_{ij}^k = 0$ in this case is still consistent with the conditions 2 and 3 in Assumption 3. Therefore, following a similar derivation as Theorem 6, we can easily get that the network will converge to the optimal solution as long as there exists an integer $B \geq 1$ such that for each $e_{ij} \in E_\infty$, $b_{i\to j}^k$ and $b_{j\to i}^k$ are both nonzero for at least once every B consecutive time slots.

Remark 8. *According to [87], when the weights a_{ij}^k are identical and time-invariant, i.e., all equal to $1/N$, the convergence rate of algorithm* (4.3)-(4.4) *is geometric, i.e., there exists a $\lambda \in (0,1)$ and some positive constant C such that $\| \boldsymbol{x}^k - \boldsymbol{x}^* \| \leq C\lambda^k$ holds for all k [88]. When a_{ij}^k are time-varying, according to [87], the convergence rate is mainly determined by the rate at which the transition matrix $\Phi(k,s) = A(s)A(s+1)...A(k-1)A^k$ converges to $\frac{1}{N}\mathbf{1}^{\mathbf{T}}\mathbf{1}$, where $\mathbf{1}$ represents a column vector of all ones and the (i,j)th entry of A^k is equal to a_{ij}^k. So next we analyze the influence of the privacy-preserving mechanism on convergence rate by analyzing the influence of random a_{ij}^k on the convergence of $\Phi(k,s)$. Note that the convergence of the transition matrix $\Phi(k,s)$ is established in [87]:*

$$|[\Phi(k,s)]_i^j - \frac{1}{N}| \leq 2\frac{1+\eta^{-B_0}}{1-\eta^{B_0}}(1-\eta^{B_0})^{(k-s)/B_0},$$

where $B_0 = (N-1)B$ with B defined in Assumption 5 and N the total number of agents. It is clear that a greater η leads to a higher convergence speed. However, from Theorem 6 we know that all a_{ij}^k should be randomly chosen in $[\eta, \frac{1-\eta}{N-1}]$, and hence to provide stronger privacy protection, η should be set smaller to ensure that weights a_{ij}^k can randomly vary in a larger range. Therefore, there is a trade-off in choosing η: a smaller η leads to stronger privacy protection but a lower convergence speed.

4.4 Privacy Analysis

In this section, we rigorously prove that each agent's private information, e.g., immediate estimate (state) $\boldsymbol{x}_j^k$ and private local objective function f_j, cannot be inferred by honest-but-curious adversaries and external eavesdroppers, which are commonly-used attack models in privacy studies [36, 62] (cf. definitions in Sec. 4.1). It is worth noting that the form of each agent's local objective function f_j can also be totally inaccessible to others, i.e., whether it is a quadratic, exponential, or other forms of convex functions is only known to an agent itself. In addition, the distribution of $b_{j\to i}$ is private to agent j itself.

As indicated in Sec. 4.3, our approach in Algorithm 3 guarantees that state information is not leaked to any neighbors in one iteration. However, would some information get leaked to an honest-but-curious adversary over time? More specifically, if an honest-but-curious adversary observes carefully its communications with neighbors over several steps, can it put together all the received information to infer its neighbor's state?

We can rigorously prove that an honest-but-curious adversary cannot infer the states of its neighbors even by collecting samples from multiple steps.

Theorem 7. *In Algorithm 3, an agent j's state $\boldsymbol{x}_j^k$ cannot be inferred by an honest-but-curious neighboring agent i.*

Proof: Suppose that an honest-but-curious agent i collects information from K iterations to infer the information of a neighboring agent j. From the perspective of the adversary agent i, the measurement (corresponding to neighboring agent j) seen in each iteration k is $\boldsymbol{y}^k = b_{i\to j}^k b_{j\to i}^k(\boldsymbol{x}_j^k - \boldsymbol{x}_i^k)$ $(k = 0, 1, 2, ..., K)$, i.e., based on received information, the adversary agent i can establish $(K+1)D$ equations with respect to the state of agent j:

$$\begin{cases} \boldsymbol{y}^0 = b_{i\to j}^0 b_{j\to i}^0(\boldsymbol{x}_j^0 - \boldsymbol{x}_i^0), \\ \boldsymbol{y}^1 = b_{i\to j}^1 b_{j\to i}^1(\boldsymbol{x}_j^1 - \boldsymbol{x}_i^1), \\ \quad\vdots \\ \boldsymbol{y}^K = b_{i\to j}^K b_{j\to i}^K(\boldsymbol{x}_j^K - \boldsymbol{x}_i^K). \end{cases} \tag{4.6}$$

To the adversary agent i, in the system of equations (4.6), $\boldsymbol{y}^k$, $b_{i\to j}^k$, $\boldsymbol{x}_i^k$ $(k = 0, 1, 2, ..., K)$ are known, but $\boldsymbol{x}_j^k, b_{j\to i}^k$ $(k = 0, 1, 2, ..., K)$ are unknown. So the above system of $(K+1)D$ equations contains $(K+1)D + K + 1$ unknown variables. It is clear that the adversary agent i cannot solve the system of equations to infer the unknowns $\boldsymbol{x}_j^k$ or $b_{j\to i}^k$ $(k = 0, 1, 2, ..., K)$ of agent j. ■

Based on a similar line of reasoning, we can obtain that an honest-but-curious agent i cannot infer the private information of function f_j from a neighboring agent j either.

Corollary 4. *In Algorithm 3, agent j's private local function f_j will not be revealed to an honest-but-curious agent i.*

Proof: Suppose that an honest-but-curious agent i collects information from K iterations to infer the function f_j of a neighboring agent j. The adversary agent i can establish KD equations with respect to f_j by making use of the fact that the update rule (4.4) is publicly known, i.e.,

$$\begin{cases} \boldsymbol{x}_j^1 = P_{\mathcal{X}}[\boldsymbol{v}_j^0 - \alpha_0 d_j^0], \\ \boldsymbol{x}_j^2 = P_{\mathcal{X}}[\boldsymbol{v}_j^1 - \alpha_1 d_j^1], \\ \quad\vdots \\ \boldsymbol{x}_j^K = P_{\mathcal{X}}[\boldsymbol{v}_j^{K-1} - \alpha_{K-1} d_j^{K-1}]. \end{cases} \tag{4.7}$$

We discuss (4.7) under two cases. Case 1): When agent j has more than one neighbor, the values of $\boldsymbol{v}_j^k, d_j^k$ $(k = 0, 1, 2, ..., K-1)$, and $\boldsymbol{x}_j^k$ $(k = 1, 2, ..., K)$ are unknown to adversary agent i. So the above system of KD equations contains $3KD$ unknown variables; Case 2): When agent j has agent i as its only neighbor, then another set of equations $\boldsymbol{v}_j^k = \boldsymbol{x}_j^k - y^k$ $(k = 1, 2, ..., K-1)$ are accessible to agent i, and hence in combination with the equations in (4.7), agent i has access to $(2K-1)D$ equations with $3KD$ unknowns. In neither case can adversary agent i infer f_j. ■

Similarly, we have that an external eavesdropper cannot infer any private information of an agent.

Corollary 5. *Every agent's intermediate states and objective functions cannot be inferred by an external eavesdropper.*

Proof: Since all exchanged messages are encrypted and that cracking the encryption is practically infeasible [36], an external eavesdropper cannot learn anything by intercepting exchanged messages. Therefore, it cannot infer any agents' intermediate states or objective functions. ■

From the above analysis, it is obvious that agent j's private information cannot be uniquely derived by adversaries. However, an honest-but-curious neighbor i can still get some range information about the state $\boldsymbol{x}_j^k$ and this estimated range will become tighter as $\boldsymbol{x}_j^k$ converges to the optimal value as $k \to \infty$ (cf. the simulation results in Fig. 4.8). We argue that this is completely unavoidable for any privacy-preserving approaches where all agents have to agree on the same final state, upon which the privacy of $\boldsymbol{x}_j^k$ disappears. In fact, this is also acknowledged in [54], which shows that the privacy of $\boldsymbol{x}_j^k$ will vanish as $k \to \infty$.

Remark 9. *It is worth noting that an adversary agent i can combine systems of equations* (4.6) *and* (4.7) *to infer the information of a neighboring agent j. However, this will not enhance the ability of adversary agent i because the combination will not change the fact that the number of unknowns is greater than the number of establishable relevant equations.*

Remark 10. *From Theorem 7, we can see that in decentralized optimization, an agent's information will not be disclosed to other agents no matter how many neighbors it has. This is in distinct difference from the average consensus problem in [75, 80, 98] where privacy cannot be protected for an agent if it has the honest-but-curious adversary as the only neighbor. This shows the disparate difference between decentralized optimization and the linear consensus problem.*

4.5 Implementation Details

In this section, we recapitulate several technical issues that have to be addressed in the implementation of Algorithm 3, which are same to the implementation of Algorithm 2.

1. In modern communication, a real number is represented by a floating point number, while encryption techniques only work for unsigned integers. To deal with this problem, we uniformly multiplied each element of the vector message $\boldsymbol{x}_i^k \in \mathbb{R}^D$ (in floating point representation) by a sufficiently large number $N_{\max}$ and round off the fractional part during the encryption to convert it to an integer. After decryption, the result is divided by $N_{\max}$. This process is conducted in each iteration and this quantization brings an error upper-bounded by $\frac{1}{N_{\max}}$.
2. As indicated in 1, encryption techniques only work for unsigned integers. In our implementation all integer values are stored in fix-length integers (i.e., long int in C) and negative values are left in 2's complement format. Encryption and intermediate computations are carried out as if the underlying data were unsigned. When the final message is decrypted, the overflown bits (bits outside the fixed length) are discarded and the remaining binary number is treated as a signed integer which is later converted back to a real value.

4.6 Application to Average Consensus

Average consensus addresses the distributed computation of the mathematical mean of participating agents' states. In recent years, it has found applications in domains as diverse as automatic control, social sciences, signal processing, robotics, and optimization [93]. In this section, we show that the average consensus problem can be formulated as a constrained optimization problem, which, in turn, can be solved using Algorithm 3 with privacy guarantee for participating agents.

Assume that participating agents have scalar states β_i for $i = 1, 2, ..., N$. Then the problem of reaching average consensus, i.e., $\bar{\beta} = \frac{1}{N}\sum\limits_{i=1}^{N}\beta_i$, on every agent, can be formulated as the following decentralized optimization problem:

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{N}\frac{1}{2}(x-\beta_i)^2 \\ \text{subject to} \quad & x \in \mathcal{X}. \end{aligned} \tag{4.8}$$

Here $\mathcal{X}$ is assumed to be large enough to contain the average consensus value $\bar{\beta}$.

Theorem 8. *A network of N agents with individual states β_i $(i = 1, 2, ..., N)$ can distributedly compute the average $\bar{\beta}$ by solving* (4.8) *using Algorithm 3 if Assumptions 4 and 5 hold, all $b_{i\to j}^k$ are randomly chosen from $[\sqrt{\eta}, \sqrt{\frac{1-\eta}{N-1}}]$ with $0 < \eta < 1/N$, and the stepsize α_k satisfies $\sum\limits_k \alpha_k = \infty$ and $\sum\limits_k \alpha_k^2 < \infty$.*

Proof: The proof can be obtained following a similar line of reasoning of Theorem 6 and hence is omitted here. ■

In addition, we have that an honest-but-curious agent i cannot infer the state of any other agents.

Theorem 9. *Agent j's private state β_j cannot be inferred by an honest-but-curious neighboring agent i if the network is composed of more than two agents, i.e., $N > 2$, and the stepsize α_k satisfies $\alpha_k \neq 1$ for all $k \geq 0$.*

Proof: The proof can be obtained following a similar line of reasoning of Theorem 7 and hence is omitted here. ■

Remark 11. *The condition $\alpha_k \neq 1$ for all $k \geq 0$ is easy to satisfy. For example, the commonly used form of $\alpha_k = \frac{T_1}{k+T_2}$ $(0 < T_1 < T_2 < \infty)$ in [17, 84] naturally satisfies this condition.*

Remark 12. *It is worth noting that in average consensus, the update rule of d_j^k, i.e., $d_j^k = v_j^k - \beta_j$, can be known to every participating agent. This is different from the general constrained optimization problem* (4.1)*, where f_j and the update rule of d_j^k are completely private to agent j. Therefore, average consensus requires a stronger condition for privacy-preservation, i.e., $N > 2$ and $\alpha_k \neq 1$ for all $k \geq 0$. However, the condition $N > 2$ is still less restrictive than the condition of requiring at least two neighbors in existing data-obfuscation based privacy-preserving average consensus results [75, 80]. In addition, as all exchanged messages are encrypted, our approach is also resilient to outside eavesdroppers, which will fail existing approaches in [75, 80].*

4.7 Numerical Simulations

In this section, we first illustrate the efficiency of the proposed privacy-preserving approach using C/C++ implementations. Then we compare our privacy-preserving average consensus approach with existing results in [80] and [75]. We used the open-source C implementation of the Paillier cryptosystem [12] in our simulations.

In the implementation, $N_{\max}$ was set to 10^6 to convert each element in $\boldsymbol{x}_i$ to a 64-bit integer during intermediate computations. $b_{i\to j}^k$ and $b_{j\to i}^k$ were also scaled up in the same way and represented by 64-bit integers. The encryption and decryption keys were chosen as 256-bit long.

4.7.1 Evaluation of Algorithm 3

We evaluated the effectiveness of Algorithm 3 using the source localization problem (cf. Example 1), agreement problem (cf. Example 2), and a least square problem (cf. Example 3), which are typical and important applications of the decentralized optimization problem (4.1).

4.7.1.1 Source Localization

We implemented Algorithm 3 under different source localization setups with sensors randomly distributed in the plane $[0,100]\times[0,100]$ and a source located at $[50;45]$. Simulation results confirmed that Algorithm 3 always converged to the source position when the source was located in the convex hull of all sensors. Fig. 4.2 visualizes the evolution of $\boldsymbol{x}_i$ $(i=1,2,3,4)$ in one specific run where the network deployment is illustrated in Fig. 4.1. In Fig. 4.2, x_{ij} $(i=1,2,3,4, j=1,2)$ denotes the jth element of $\boldsymbol{x}_i$. All $\boldsymbol{x}_i$ $(i=1,2,3,4)$ converged to the source position $[50;45]$. Fig. 4.3 visualizes the encrypted weighted differences (in ciphertext) $\mathcal{E}_1(b_{2\to1}^k(\boldsymbol{x}_{21}^k-\boldsymbol{x}_{11}^k))$, $\mathcal{E}_1(b_{3\to1}^k(\boldsymbol{x}_{31}^k-\boldsymbol{x}_{11}^k))$, and $\mathcal{E}_1(b_{4\to1}^k(\boldsymbol{x}_{41}^k-\boldsymbol{x}_{11}^k))$. It is worth noting that although the estimates of all agents have converged after about 30 iterations, the encrypted weighted differences (in ciphertext) still appeared random to an outside eavesdropper. For Paillier cryptosystem, if the encryption/decryption key-length is n-bit, the size of ciphertexts will be $2n$ [20]. Since we use 256-bit key, the ciphertext is 512-bit, i.e. as large as 2^{512}.

4.7.1.2 Agreement Problem

We next implemented Algorithm 3 under different network topologies to solve the agreement problem in Example 2. Simulation results confirmed that Algorithm 3 always converged to the optimal solution $\frac{\sum_{i=1}^{N}\boldsymbol{\theta}_i}{N}$.

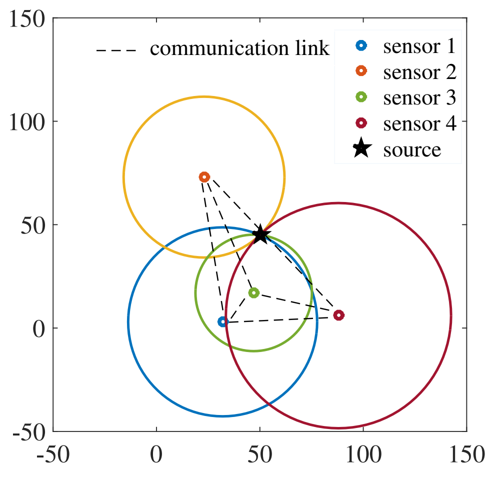


Figure 4.1: Source localization setup used in one simulation run.

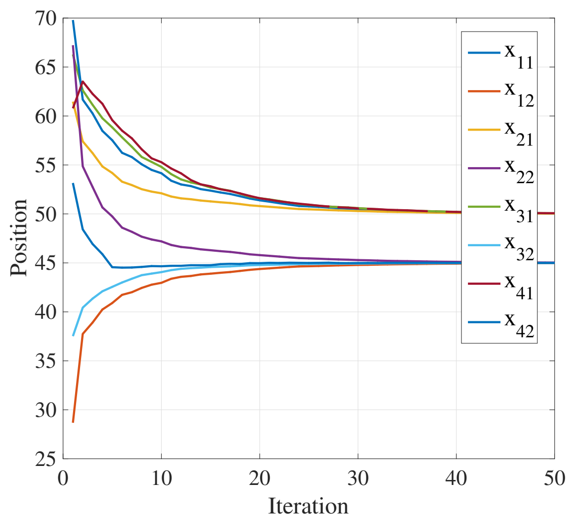


Figure 4.2: The evolution of $\boldsymbol{x}_i$ ($i = 1, 2, 3, 4$) in Algorithm 3.

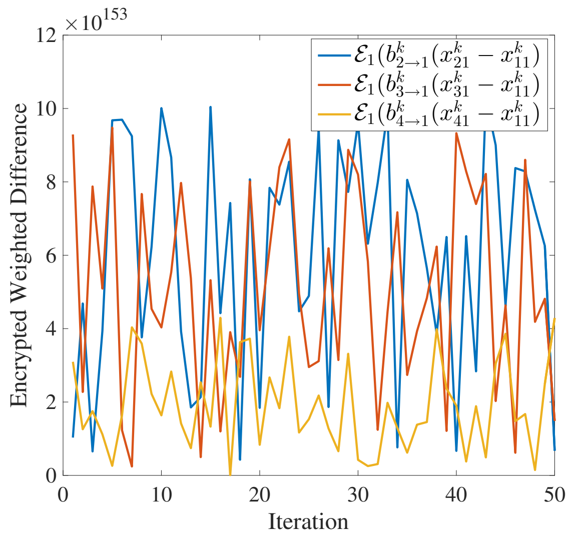


Figure 4.3: The evolution of the encrypted wighted differences (in ciphertext) $\mathcal{E}_1(b^k_{2\to1}(\boldsymbol{x}^k_{21} - \boldsymbol{x}^k_{11}))$, $\mathcal{E}_1(b^k_{3\to1}(\boldsymbol{x}^k_{31} - \boldsymbol{x}^k_{11}))$, and $\mathcal{E}_1(b^k_{4\to1}(\boldsymbol{x}^k_{41} - \boldsymbol{x}^k_{11}))$.

Fig. 4.5 visualizes the evolution of $\boldsymbol{x}_i$ $(i = 1, 2, ..., 6)$ in one specific run where the network communication topology is given in Fig. 4.4 and α_k was set to $\alpha_k = \frac{1}{k+2}$. In Fig. 4.5, x_{ij} $(i = 1, 2, ..., 6, j = 1, 2)$ denotes the jth element of $\boldsymbol{x}_i$. All $\boldsymbol{x}_i$ $(i = 1, 2, ..., 6)$ converged to the optimal value $[48.5; \frac{373}{6}]$.

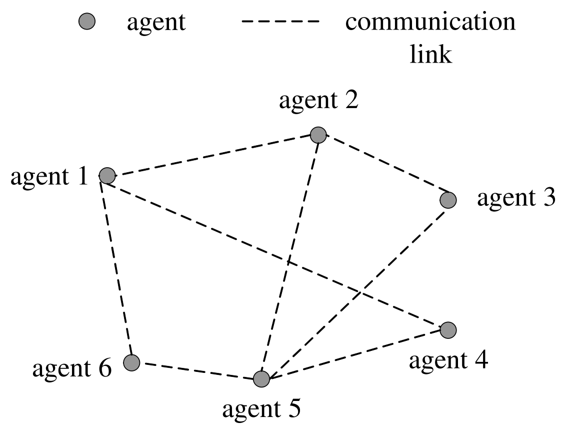


Figure 4.4: A network of six agents ($N = 6$).

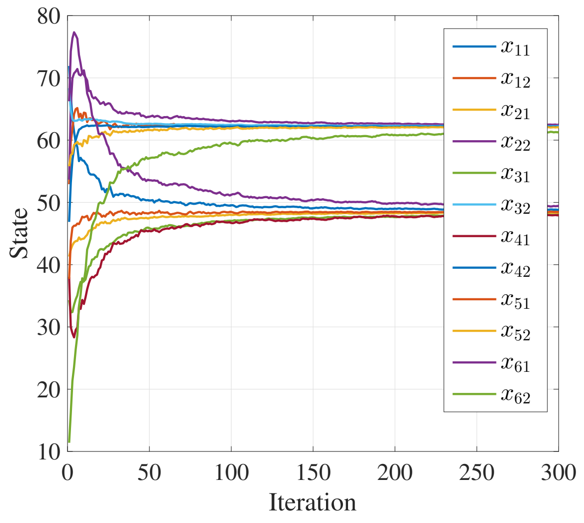


Figure 4.5: The evolution of $\boldsymbol{x}_i$ $(i = 1, 2, ..., 6)$ in Algorithm 3.

4.7.1.3 Least Squares Problem

We also implemented Algorithm 3 to solve the simple linear regression problem in Example 3. Simulation results confirmed that Algorithm 3 always converged to the optimal solution. However, the convergence speed was lower than the source localization problem and the agreement problem. Simulation results also suggested that the convergence rate was sensitive to the stepsize α_k. Fig. 4.6 visualizes the evolution of γ_0^i and γ_1^i $(i = 1, 2, ..., 6)$ in one specific run where the network communication topology is given in Fig. 4.4, α_k was set to $\alpha_k = \frac{10}{k+20}$, x_i was set to $x_i = i$, y_i was set to $y_i = 2 \times i - 16$, and the constrained set was set to $\gamma_0^2 + \gamma_1^2 \leq 500^2$. γ_0^i and γ_1^i $(i = 1, 2, ..., 6)$ denote the intermediate states γ_0 and γ_1 of agent i, respectively. All γ_0^i and γ_1^i $(i = 1, 2, ..., 6)$ converged to their respective optimal values, i.e., 2 and -16.

4.7.1.4 The Effect of Encryption and Decryption Key-length and Network Size

We also considered the influence of encryption and decryption key-length and network size on Algorithm 3 (based on the agreement problem). We simulated two all-to-all networks with 6 and 51 agents, respectively. Both 256-bit and 2048-bit keys are evaluated. Table 4.1 gives the average computation time of encryption and decryption for each agent to communicate with all its neighbors in each iteration on a 3.6 GHz CPU with 15.6 GB RAM. Fig. 4.7 visualizes the evolution of $\boldsymbol{x}_i$ $(i = 1, 2, ..., 51)$ in the all-to-all network with 51 agents using 2048-bit keys. We can see that the average computation time increased with increased key-length and network size.

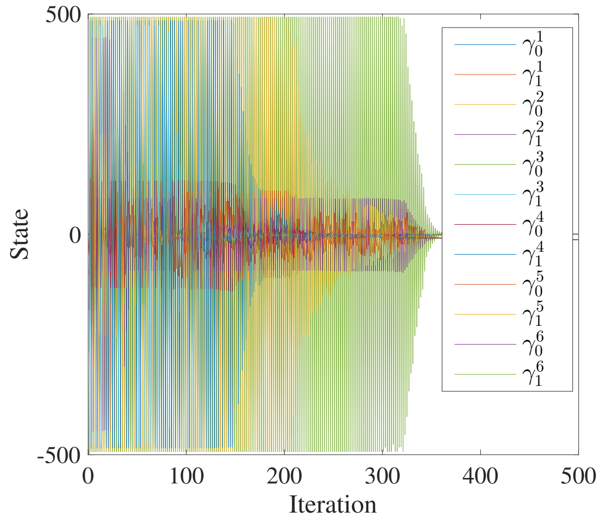


Figure 4.6: The evolution of γ_0^i and γ_1^i $(i = 1, 2, ..., 6)$ in Algorithm 3.

Table 4.1: Average encryption and decryption computation time

network size	key-length (bit)	average time (s)
6	256	0.005
	2048	0.193
51	256	0.042
	2048	1.763

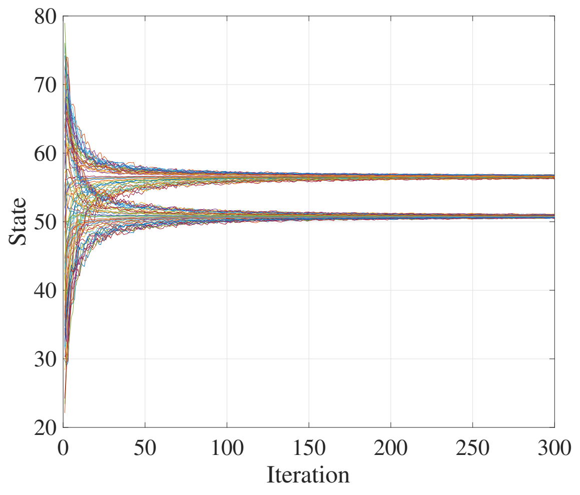


Figure 4.7: The evolution of $\boldsymbol{x}_i$ $(i = 1, 2, ..., 51)$ in Algorithm 3.

4.7.1.5 The Trade-off between Convergence Speed and Privacy

In this part, we first simulated an honest-but-curious adversary who tried to estimate its neighbors' intermediate states under different η values to illustrate the strengths of enabled privacy under different values of η. Then we simulated the convergence speed of Algorithm 3 under different η values. All simulation results were obtained based on the agreement problem.

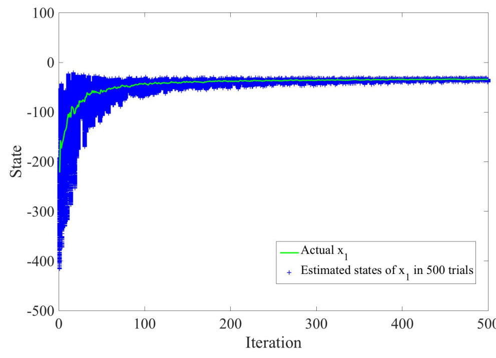
Assume that agent 2 in Fig. 4.4 is an honest-but-curious adversary who intends to estimate the intermediate states of agent 1. The individual local objective functions are the same as in (4.2) with $\theta_i \in \mathbb{R}$. Because agent 2 knows the constraints on agent 1's generation of $b_{1\to 2}$, i.e., $b_{1\to 2}$ is randomly chosen from $[\sqrt{\eta}, \sqrt{\frac{1-\eta}{N-1}}]$ with $0 < \eta < 1/N$ (cf. Theorem 6), it generated estimates of $b_{1\to 2}$ by using a guessed stochastic distribution of $b_{1\to 2}$. We conservatively assume that agent 2 knows the probability distribution of $b_{1\to 2}$, which gives it an edge in estimating $b_{1\to 2}$. Then agent 2 obtained a series of estimated x_1^k according to (4.6). For example, after agent 2 obtained $y^k = b_{2\to 1}^k b_{1\to 2}^k (x_1^k - x_2^k)$ at iteration k, it generated an estimate of $b_{1\to 2}^k$ (denoted as $\bar{b}_{1\to 2}^k$), and then it estimated x_1^k as $x_1^k = x_2^k + \frac{y^k}{\bar{b}_{1\to 2}^k b_{2\to 1}^k}$.

Fig. 4.8 shows the estimated x_1 in 500 trials under different η values when $b_{1\to 2}$ follows uniform distribution. It can be seen that a smaller η leads to less accurate estimation and hence better privacy protection, confirming the statement in Remark 8. In addition, it can be seen that agent 2 cannot accurately estimate x_1 initially. However, as x_1 converges to the optimal value, agent 2 will be able to estimate the value that every agent agrees on, confirming the statement right above Remark 9.

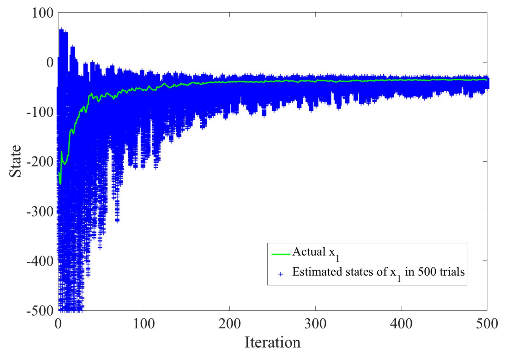
We use the root mean square error (RMSE) to quantify the error between intermediate states and the optimal value, which is denoted as $\mathrm{ERR}_{\mathrm{RMSE}}$:

$$\mathrm{ERR}_{\mathrm{RMSE}} = \sqrt{\frac{\sum\limits_{j=1}^{L}\sum\limits_{i=1}^{N} \| \boldsymbol{x}_{ij} - \boldsymbol{x}^* \|^2}{LN}},$$

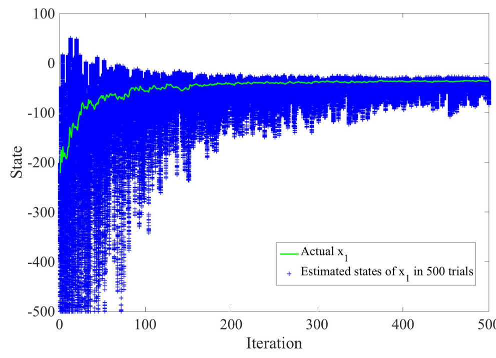
where L is the number of Monte Carlo trials, N is the number of agents, $\boldsymbol{x}_{ij}$ is the intermediate state of agent i in the jth Monte Carlo trial, and $\boldsymbol{x}^*$ is the optimal value. Fig. 4.9(a) visualizes the evolution of $\mathrm{ERR}_{\mathrm{RMSE}}$ under different η values when the network topology is given in Fig. 4.4 ($L = 500$) and Fig. 4.9(b) visualizes the evolution of $\mathrm{ERR}_{\mathrm{RMSE}}$ under different η values in an all-to-all network with 51 agents ($L = 500$). It can be seen that to reach the same $\mathrm{ERR}_{\mathrm{RMSE}}$, a smaller η incurs more iterations for convergence, confirming the statement in Remark 8. Because our approach requires a small η to enable strong privacy protection, it sacrifices the convergence speed in this sense.



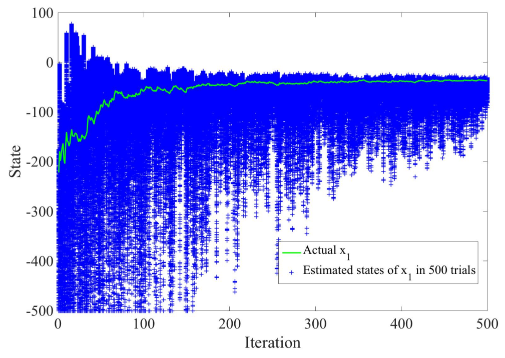
(a) $\eta = 0.05$



(b) $\eta = 0.01$

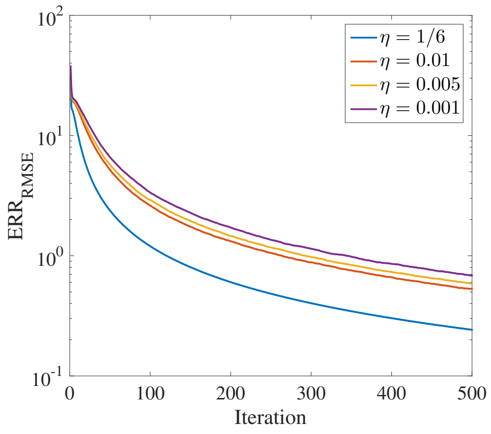


(c) $\eta = 0.005$

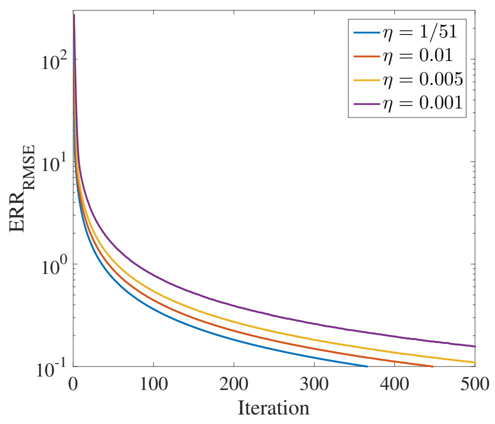


(d) $\eta = 0.001$

Figure 4.8: An adversary's estimation of the intermediate state of agent 1. The green line is the actual intermediate state x_1 of agent 1, the blue "+" are estimated states of x_1 by agent 2 in 500 trials.



(a) The evolution of ERR_{RMSE} under different η values when the network topology is given in Fig. 4.4.



(b) The evolution of ERR_{RMSE} under different η values in an all-to-all network with 51 agents.

Figure 4.9: The evolution of ERR_{RMSE} under different η values in Algorithm 3.

4.7.2 Privacy-preserving average consensus

Using the network communication topology in Fig. 4.4, we compared our privacy-preserving average consensus approach with the algorithms in [80] and [75]. We set the states β_i of the six agents to $\{1, 2, 3, 4, 5, 6\}$ respectively. The weights were set as follows:

$$a_{ij} = \begin{cases} 0.2 & j \in \mathcal{N}_i, \\ 0 & j \notin \mathcal{N}_i \cup \{i\}, \\ 1 - \sum\limits_{j \in \mathcal{N}_i} a_{ij} & i = j, \end{cases} \tag{4.9}$$

The internal state and the exchanged state are denoted as x_i^k and x_i^{+k} for the algorithms in [80] and [75].

Fig. 4.10 visualizes the evolution of x_i $(i = 1, 2, ..., 6)$ under the proposed approach in one specific run where α_k was set to $\alpha_k = \frac{1}{k+2}$. It can be see than all x_i converged to the exact average value 3.5, confirming the effectiveness of the proposed approach.

It is worth noting that the convergence speed of the privacy-preserving average consensus approach can be increased by judiciously designing the stepsize α_k. For example, simulation results suggested that using the α_k below, convergence to the average can be made much faster (cf. the evolution of x_i in Fig. 4.11).

$$\alpha_k = \begin{cases} \dfrac{10}{k+20} & k < 30, \\ \dfrac{1}{k+1000} & k \geq 30. \end{cases} \tag{4.10}$$

Since our approach encrypts all exchanged messages, an outside eavesdropper cannot learn anything by intercepting these messages. In contrast, the algorithms in [80] and [75] cannot protect the privacy of participating agents against an external eavesdropper that can intercept all exchanged messages, as confirmed by our numerical simulation results below. Without loss of generality, we assume that an outside eavesdropper is interested in learning the state β_1 of agent 1 and builds the following observer to estimate β_1:

$$z^{k+1} = z^k + x_1^{+(k+1)} - (a_{11}x_1^{+k} + \sum_{j \in \mathcal{N}_1} a_{1j}x_j^{+k}) \tag{4.11}$$

with the initial value of z set to $z^0 = x_1^{+0}$. As mentioned earlier, in (4.11) x_1 and x_1^+ denote the internal and exchanged states, respectively. Fig. 4.12 visualizes the evolution of x_i $(i = 1, 2, ..., 6)$ as well as the

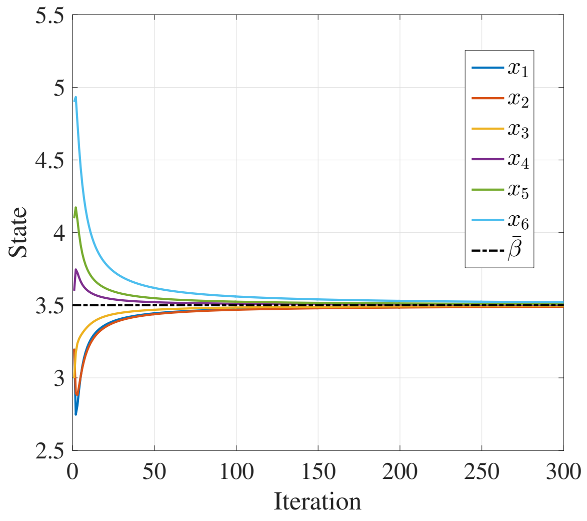


Figure 4.10: The evolution of x_i ($i = 1, 2, ..., 6$) under the proposed privacy-preserving average consensus approach when α_k was set to $\alpha_k = \frac{1}{k+2}$ in Algorithm 3.

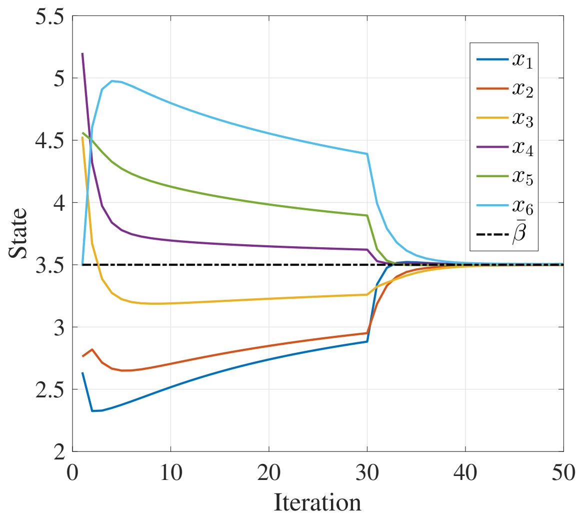


Figure 4.11: The evolution of x_i ($i = 1, 2, ..., 6$) under the proposed privacy-preserving average consensus approach when α_k was set according to (4.10) in Algorithm 3.

eavesdropper's observer state z^k under the approach in [80]. It can be seen that the eavesdropper can accurately estimate the internal state x_1. The same conclusion can be drawn for the approach in [75], which is confirmed vulnerable to eavesdropping attacks (cf. Fig. 4.13).

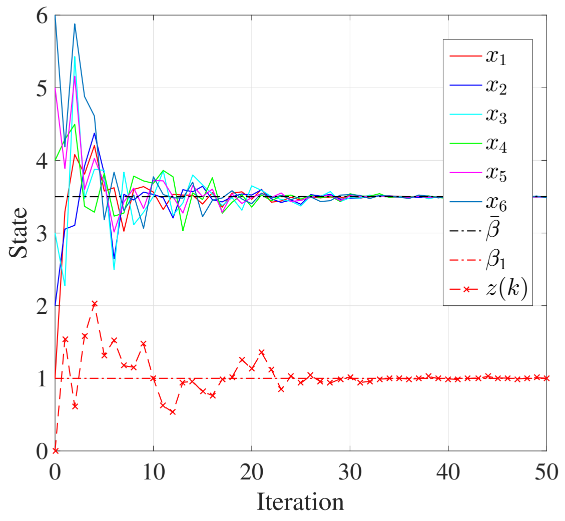


Figure 4.12: The evolution of x_i ($i = 1, 2, ..., 6$) and z^k under the algorithm in [80].

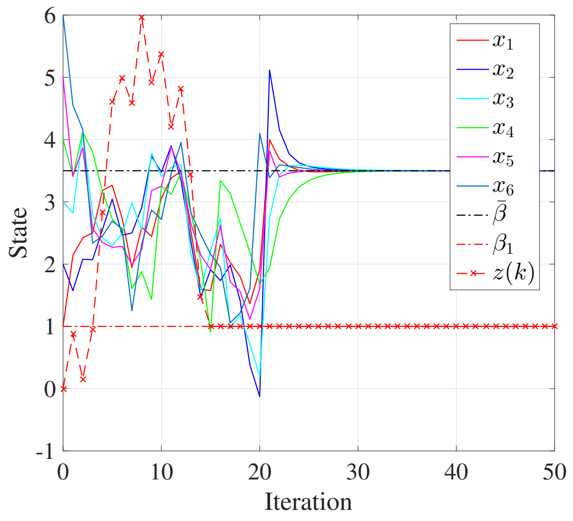


Figure 4.13: The evolution of x_i ($i = 1, 2, ..., 6$) and z^k under the algorithm in [75].

4.8 Summaries

In this chapter, we proposed a novel approach to enabling privacy-preservation in decentralized optimization based on the integration of partially homomorphic cryptography with subgradient method. By leveraging Paillier cryptosystem and the convergence properties of subgradient method, i.e., robustness to random coupling weights, our approach provides privacy guarantee without compromising the optimality of optimization in the absence of an aggregator or third party. Theoretical analysis confirms that an honest-but-curious adversary cannot infer the information of neighboring agents even by recording and analyzing the information exchanged in multiple iterations. The approach is also applicable to average consensus which has found extensive applications in fields as diverse as distributed computing, robotic networks, and power grids. Numerical simulation results confirmed the effectiveness and low computational complexity of the proposed approach.

Chapter 5

Privacy-preserving Decentralized Optimization using Function Decomposition

This chapter proposes a function-decomposition based privacy-preserving approach for the decentralized optimization problem (3.1) using Jacobian ADMM. Compared with encryption-based approaches which suffer from heavy computational and communication burden, the proposed approach incurs little extra computational and communication overhead. We also prove that when the global objective function is strongly convex, proximal Jacobian ADMM can achieve Q-linear convergence rate[1] even when local individual objective functions are only convex, which generalizes existing results on proximal Jacobian ADMM requiring strongly convex local objective functions to achieve Q-linear convergence rate.

In this chapter, we also consider the two types of adversaries defined in 3.1, which are *Honest-but-curious adversaries* [36, 62] and *External eavesdroppers*. In addition, we define privacy as preserving the confidentiality of agents' objective functions in this chapter.

The rest of this Chapter is organized as follows: Sec. 5.1 presents the proximal Jacobian ADMM solution to (3.1). Then a completely decentralized privacy-preserving approach to problem (3.1) is proposed in Sec. 5.2. Rigorous analysis of the guaranteed privacy and convergence is addressed in Sec. 5.3 and Sec.

[1]For a sequence $\{\boldsymbol{x}^k\}$ converging to $\boldsymbol{x}^*$ in some norm, Q-linear convergence rate is achieved if there exists a $\lambda \in (0,1)$ such that $\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \| \leq \lambda \| \boldsymbol{x}^k - \boldsymbol{x}^* \|$ holds for all k [85].

5.4, respectively. Numerical simulation results are provided in Sec. 5.5 to confirm the effectiveness of the proposed approach. In the end, we draw summaries in Sec. 5.6.

5.1 Background

The decentralized problem (3.1) can be formulated as follows: each f_i in (3.1) is private and only known to agent i, and all N agents form a bidirectional connected network, which is denoted by a graph $G = (V, E)$. V denotes the set of agents, E denotes the set of communication links (undirected edges) between agents, and $|E|$ denotes the number of communication links (undirected edges) in E. If there exists a communication link between agents i and j, we say that agent i and agent j are neighbors and the link is denoted as $e_{i,j} \in E$ if $i < j$ is true or $e_{j,i} \in E$ otherwise. Moreover, the set of all neighboring agents of i is denoted as $\mathcal{N}_i$ and the number of agents in $\mathcal{N}_i$ is denoted as N_i. Then problem (3.1) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{x}_i \in \mathbb{R}^D,\, i \in \{1,2,\ldots,N\}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{x}_j, \quad \forall e_{i,j} \in E, \end{aligned} \tag{5.1}$$

where $\boldsymbol{x}_i$ is a copy of $\tilde{\boldsymbol{x}}$ belonging to agent i.

In the conventional proximal Jacobian ADMM [26], each agent uses the following update to cooperatively find the optimal solution to (3.1):

$$\begin{cases} \boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} f_i(\boldsymbol{x}_i) + \frac{\gamma_i \rho}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^k \parallel^2 + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{i,j}^{kT}(\boldsymbol{x}_i - \boldsymbol{x}_j^k) + \frac{\rho}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j^k \parallel^2) & (5.2) \\ \boldsymbol{\lambda}_{i,j}^{k+1} = \boldsymbol{\lambda}_{i,j}^k + \rho\tau(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^{k+1}), \quad \forall j \in \mathcal{N}_i & (5.3) \end{cases}$$

Here, k is the iteration index, $\gamma_i > 0$ $(i = 1, 2, \ldots, N)$ are proximal coefficients, $\tau > 0$ is a damping parameter, ρ is the penalty parameter, which is a positive constant scalar. $\boldsymbol{\lambda}_{i,j}$ and $\boldsymbol{\lambda}_{j,i}$ are Lagrange multipliers corresponding to the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j, e_{i,j} \in E$. Here, both $\boldsymbol{\lambda}_{i,j}$ and $\boldsymbol{\lambda}_{j,i}$ are introduced for the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j, e_{i,j} \in E$ in (5.2)-(5.3) to unify the algorithm description. By setting $\boldsymbol{\lambda}_{i,j}^0 = \rho(\boldsymbol{x}_i^0 - \boldsymbol{x}_j^0)$ at $t = 0$, we have $\boldsymbol{\lambda}_{i,j}^k = -\boldsymbol{\lambda}_{j,i}^k$ for all $i = 1, 2, \cdots, N, j \in \mathcal{N}_i$. In this way, we unify the update rule of agent i without separating $i > j$ and $i < j$ for $j \in \mathcal{N}_i$, as is clear in (5.2).

The conventional proximal Jacobian ADMM is effective in solving (3.1). However, it cannot protect the privacy of participating agents' gradients as states $\boldsymbol{x}_i^k$ are exchanged and disclosed explicitly among neighboring agents. Adversaries can easily derive $\triangledown f_i(\boldsymbol{x}_i^k)$ using the update rules in (5.2) and (5.3) by

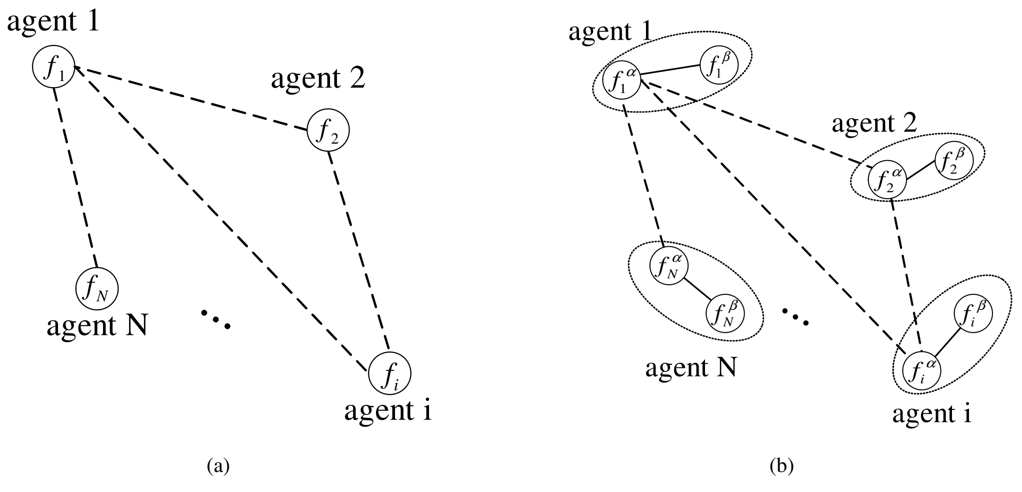


Figure 5.1: Function-decomposition based privacy-preserving decentralized optimization. (a) Before function decomposition. (b) After function decomposition.

leveraging the knowledge of γ_i.

5.2 Privacy-preserving Decentralized Optimization

The key idea of our approach to enabling privacy-preservation is to randomly decompose each f_i into two parts $f_i^{\alpha k}$ and $f_i^{\beta k}$ under the constraint $f_i = f_i^{\alpha k} + f_i^{\beta k}$. The index k of functions $f_i^{\alpha k}$ and $f_i^{\beta k}$ indicates that functions $f_i^{\alpha k}$ and $f_i^{\beta k}$ can be time-varying. However, it should be noticed that the sum of $f_i^{\alpha k}$ and $f_i^{\beta k}$ is time invariant and always equals to f_i. We let the function $f_i^{\alpha k}$ succeed the role of the original function f_i in inter-agent interactions while the other function $f_i^{\beta k}$ involves only by interacting with $f_i^{\alpha k}$, as shown in Fig. 5.1.

After the function decomposition, problem (3.1) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{x}_i^\alpha, \boldsymbol{x}_i^\beta \in \mathbb{R}^D,\, i \in \{1,2,\ldots,N\}} \quad & \sum_{i=1}^{N} (f_i^{\alpha k}(\boldsymbol{x}_i^\alpha) + f_i^{\beta k}(\boldsymbol{x}_i^\beta)) \\ \text{subject to} \quad & \boldsymbol{x}_i^\alpha = \boldsymbol{x}_j^\alpha, \quad \forall e_{i,j} \in E, \\ & \boldsymbol{x}_i^\alpha = \boldsymbol{x}_i^\beta, \quad \forall i \in V, \end{aligned} \tag{5.4}$$

and the associated augmented Lagrangian function is:

$$\begin{aligned}\mathcal{L}_\rho^k(\boldsymbol{x},\boldsymbol{\lambda}) = &\sum_{i=1}^{N}(f_i^{\alpha k}(\boldsymbol{x}_i^\alpha)+f_i^{\beta k}(\boldsymbol{x}_i^\beta)) + \sum_{e_{i,j}\in E}(\boldsymbol{\lambda}_{i,j}^{\alpha T}(\boldsymbol{x}_i^\alpha-\boldsymbol{x}_j^\alpha)+\frac{\rho}{2}\parallel \boldsymbol{x}_i^\alpha-\boldsymbol{x}_j^\alpha\parallel^2)\\ &+\sum_{i\in V}(\boldsymbol{\lambda}_{i,i}^{\alpha\beta T}(\boldsymbol{x}_i^\alpha-\boldsymbol{x}_i^\beta)+\frac{\rho}{2}\parallel \boldsymbol{x}_i^\alpha-\boldsymbol{x}_i^\beta\parallel^2),\end{aligned} \tag{5.5}$$

where $\boldsymbol{x}=[\boldsymbol{x}_1^{\alpha T},\boldsymbol{x}_1^{\beta T},\boldsymbol{x}_2^{\alpha T},\boldsymbol{x}_2^{\beta T},\ldots,\boldsymbol{x}_N^{\alpha T},\boldsymbol{x}_N^{\beta T}]^T\in\mathbb{R}^{2DN}$ is the augmented state. $\boldsymbol{\lambda}_{i,j}^\alpha$ is the Lagrange multiplier corresponding to the constraint $\boldsymbol{x}_i^\alpha=\boldsymbol{x}_j^\alpha$, $\boldsymbol{\lambda}_{i,i}^{\alpha\beta}$ is the Lagrange multiplier corresponding to the constraint $\boldsymbol{x}_i^\alpha=\boldsymbol{x}_i^\beta$, and all $\boldsymbol{\lambda}_{i,j}^\alpha$ and $\boldsymbol{\lambda}_{i,i}^{\alpha\beta}$ are stacked into $\boldsymbol{\lambda}$. ρ is the penalty parameter, which is a positive constant scalar. It is worth noting that agent i does not need to know the associated augmented Lagrangian function (i.e., other agents' objective functions) to update its states $\boldsymbol{x}_i^\alpha$ and $\boldsymbol{x}_i^\beta$, as shown below in (5.6) and (5.7).

Based on Jacobian update, we can solve (5.4) by applying the following iterations for $i=1,2,\ldots,N$:

$$\begin{cases}\boldsymbol{x}_i^{\alpha(k+1)} = \underset{\boldsymbol{x}_i^\alpha}{\operatorname{argmin}}\frac{\gamma_i^\alpha\rho}{2}\parallel\boldsymbol{x}_i^\alpha-\boldsymbol{x}_i^{\alpha k}\parallel^2+\mathcal{L}_\rho^{k+1}(\boldsymbol{x}_1^{\alpha k},\boldsymbol{x}_1^{\beta k},\ldots,\boldsymbol{x}_i^{\alpha},\boldsymbol{x}_i^{\beta k},\ldots,\boldsymbol{x}_N^{\alpha k},\boldsymbol{x}_N^{\beta k},\boldsymbol{\lambda}^k)\\ \quad = \underset{\boldsymbol{x}_i^\alpha}{\operatorname{argmin}} f_i^{\alpha(k+1)}(\boldsymbol{x}_i^\alpha)+\frac{\gamma_i^\alpha\rho}{2}\parallel\boldsymbol{x}_i^\alpha-\boldsymbol{x}_i^{\alpha k}\parallel^2+\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^{\alpha kT}(\boldsymbol{x}_i^\alpha-\boldsymbol{x}_j^{\alpha k})+\frac{\rho}{2}\parallel\boldsymbol{x}_i^\alpha-\boldsymbol{x}_j^{\alpha k}\parallel^2)\\ \quad +\boldsymbol{\lambda}_{i,i}^{\alpha\beta kT}(\boldsymbol{x}_i^\alpha-\boldsymbol{x}_i^{\beta k})+\frac{\rho}{2}\parallel\boldsymbol{x}_i^\alpha-\boldsymbol{x}_i^{\beta k}\parallel^2, & (5.6)\\ \boldsymbol{x}_i^{\beta(k+1)} = \underset{\boldsymbol{x}_i^\beta}{\operatorname{argmin}}\frac{\gamma_i^\beta\rho}{2}\parallel\boldsymbol{x}_i^\beta-\boldsymbol{x}_i^{\beta k}\parallel^2+\mathcal{L}_\rho^{k+1}(\boldsymbol{x}_1^{\alpha k},\boldsymbol{x}_1^{\beta k},\ldots,\boldsymbol{x}_i^{\alpha k},\boldsymbol{x}_i^{\beta},\ldots,\boldsymbol{x}_N^{\alpha k},\boldsymbol{x}_N^{\beta k},\boldsymbol{\lambda}^k)\\ \quad = \underset{\boldsymbol{x}_i^\beta}{\operatorname{argmin}} f_i^{\beta(k+1)}(\boldsymbol{x}_i^\beta)+\frac{\gamma_i^\beta\rho}{2}\parallel\boldsymbol{x}_i^\beta-\boldsymbol{x}_i^{\beta k}\parallel^2+\boldsymbol{\lambda}_{i,i}^{\beta\alpha kT}(\boldsymbol{x}_i^\beta-\boldsymbol{x}_i^{\alpha k})+\frac{\rho}{2}\parallel\boldsymbol{x}_i^\beta-\boldsymbol{x}_i^{\alpha k}\parallel^2, & (5.7)\\ \boldsymbol{\lambda}_{i,j}^{\alpha(k+1)} = \boldsymbol{\lambda}_{i,j}^{\alpha k}+\tau\rho(\boldsymbol{x}_i^{\alpha(k+1)}-\boldsymbol{x}_j^{\alpha(k+1)}),\quad \forall j\in\mathcal{N}_i & (5.8)\\ \boldsymbol{\lambda}_{i,i}^{\alpha\beta(k+1)} = \boldsymbol{\lambda}_{i,i}^{\alpha\beta k}+\tau\rho(\boldsymbol{x}_i^{\alpha(k+1)}-\boldsymbol{x}_i^{\beta(k+1)}), & (5.9)\\ \boldsymbol{\lambda}_{i,i}^{\beta\alpha(k+1)} = \boldsymbol{\lambda}_{i,i}^{\beta\alpha k}+\tau\rho(\boldsymbol{x}_i^{\beta(k+1)}-\boldsymbol{x}_i^{\alpha(k+1)}). & (5.10)\end{cases}$$

Here $\tau\in(0,1)$ is a damping parameter, and both $\boldsymbol{\lambda}_{i,j}^\alpha$ and $\boldsymbol{\lambda}_{j,i}^\alpha$ are introduced for the constraint $\boldsymbol{x}_i^\alpha=\boldsymbol{x}_j^\alpha, e_{i,j}\in E$ in (5.6)-(5.10) to unify the algorithm description. Similarly, both $\boldsymbol{\lambda}_{i,i}^{\alpha\beta}$ and $\boldsymbol{\lambda}_{i,i}^{\beta\alpha}$ are introduced for the constraint $\boldsymbol{x}_i^\alpha=\boldsymbol{x}_i^\beta$ in (5.6)-(5.10) to unify the algorithm description. Our privacy-preserving function-decomposition based algorithm is given in Algorithm 4.

Remark 13. *Different from [65] which also considers dynamic decentralized optimization, our dynamics are added purposely to enable privacy-preservation. In addition, we use Jacobian update instead of introducing*

Algorithm 4

Initial Setup: For all $i = 1, 2, \dots, N$, agent i initializes $\boldsymbol{x}_i^{\alpha 0}$ and $\boldsymbol{x}_i^{\beta 0}$, and exchanges $\boldsymbol{x}_i^{\alpha 0}$ with neighboring agents. Then agent i sets $\boldsymbol{\lambda}_{i,j}^{\alpha 0} = \boldsymbol{x}_i^{\alpha 0} - \boldsymbol{x}_j^{\alpha 0}$, $\boldsymbol{\lambda}_{i,i}^{\alpha\beta 0} = \boldsymbol{x}_i^{\alpha 0} - \boldsymbol{x}_i^{\beta 0}$, and $\boldsymbol{\lambda}_{i,i}^{\beta\alpha 0} = \boldsymbol{x}_i^{\beta 0} - \boldsymbol{x}_i^{\alpha 0}$.

Input: $\boldsymbol{x}_i^{\alpha k}$, $\boldsymbol{\lambda}_{i,j}^{\alpha k}$, $\boldsymbol{\lambda}_{i,i}^{\alpha\beta k}$, $\boldsymbol{x}_i^{\beta k}$, $\boldsymbol{\lambda}_{i,i}^{\beta\alpha k}$.

Output: $\boldsymbol{x}_i^{\alpha(k+1)}$, $\boldsymbol{\lambda}_{i,j}^{\alpha(k+1)}$, $\boldsymbol{\lambda}_{i,i}^{\alpha\beta(k+1)}$, $\boldsymbol{x}_i^{\beta(k+1)}$, $\boldsymbol{\lambda}_{i,i}^{\beta\alpha(k+1)}$.

```
1. For all $i = 1, 2, \dots, N$, agent $i$ constructs $f_i^{\alpha(k+1)}$ and $f_i^{\beta(k+1)}$ under the constraint $f_i = f_i^{\alpha(k+1)} + f_i^{\beta(k+1)}$;
2. For all $i = 1, 2, \dots, N$, agent $i$ updates $\boldsymbol{x}_i^{\alpha(k+1)}$ and $\boldsymbol{x}_i^{\beta(k+1)}$ according to the update rules in (5.6) and (5.7), respectively;
3. For all $i = 1, 2, \dots, N$, agent $i$ sends $\boldsymbol{x}_i^{\alpha(k+1)}$ to neighboring agents;
4. For all $i = 1, 2, \dots, N$, agent $i$ computes $\boldsymbol{\lambda}_{i,j}^{\alpha(k+1)}$, $\boldsymbol{\lambda}_{i,i}^{\alpha\beta(k+1)}$ and $\boldsymbol{\lambda}_{i,i}^{\beta\alpha(k+1)}$ according to (5.8)-(5.10);
5. Set $k$ to $k+1$, and go to 1.
```

splitting variables which increases the number of variables and constraints of the problem.

5.3 Privacy Analysis

In this section, we rigorously prove that each agent's gradient of local objective function $\triangledown f_j$ cannot be inferred by honest-but-curious adversaries and external eavesdroppers.

Theorem 10. *In Algorithm 4, agent j's gradient of local objective function $\triangledown f_j$ at any point except the optimal solution will not be revealed to an honest-but-curious agent i.*

Proof. Suppose that an honest-but-curious adversary agent i collects information from K iterations to infer the gradient $\triangledown f_j$ of a neighboring agent j. The adversary agent i can establish $2DK$ equations relevant to $\triangledown f_j$

by making use of the fact that the update rules of (5.6) and (5.7) are publicly known, i.e.,

$$\begin{cases} \nabla f_j^{\alpha 1}(\boldsymbol{x}_j^{\alpha 1}) + (\gamma_j^{\alpha} + N_j + 1)\rho \boldsymbol{x}_j^{\alpha 1} - \gamma_j^{\alpha}\rho \boldsymbol{x}_j^{\alpha 0} + \sum_{m\in\mathcal{N}_j} (\boldsymbol{\lambda}_{j,m}^{\alpha 0} - \rho \boldsymbol{x}_m^{\alpha 0}) + \boldsymbol{\lambda}_{j,j}^{\alpha\beta 0} - \rho \boldsymbol{x}_j^{\beta 0} = \boldsymbol{0} \\ \nabla f_j^{\beta 1}(\boldsymbol{x}_j^{\beta 1}) + (\gamma_j^{\beta} + 1)\rho \boldsymbol{x}_j^{\beta 1} - \gamma_j^{\beta}\rho \boldsymbol{x}_j^{\beta 0} + \boldsymbol{\lambda}_{j,j}^{\beta\alpha 0} - \rho \boldsymbol{x}_j^{\alpha 0} = \boldsymbol{0} \\ \qquad\qquad\vdots \\ \nabla f_j^{\alpha K}(\boldsymbol{x}_j^{\alpha K}) + (\gamma_j^{\alpha} + N_j + 1)\rho \boldsymbol{x}_j^{\alpha K} - \gamma_j^{\alpha}\rho \boldsymbol{x}_j^{\alpha (K-1)} \\ \qquad + \sum_{m\in\mathcal{N}_j} (\boldsymbol{\lambda}_{j,m}^{\alpha (K-1)} - \rho \boldsymbol{x}_m^{\alpha (K-1)}) + \boldsymbol{\lambda}_{j,j}^{\alpha\beta (K-1)} - \rho \boldsymbol{x}_j^{\beta (K-1)} = \boldsymbol{0} \\ \nabla f_j^{\beta K}(\boldsymbol{x}_j^{\beta K}) + (\gamma_j^{\beta} + 1)\rho \boldsymbol{x}_j^{\beta K} - \gamma_j^{\beta}\rho \boldsymbol{x}_j^{\beta (K-1)} + \boldsymbol{\lambda}_{j,j}^{\beta\alpha (K-1)} - \rho \boldsymbol{x}_j^{\alpha (K-1)} = \boldsymbol{0} \end{cases} \tag{5.11}$$

In the system of $2DK$ equations (5.11), $\nabla f_j^{\alpha k}(\boldsymbol{x}_j^{\alpha k})$ $(k = 1, 2, \ldots, K)$, $\nabla f_j^{\beta k}(\boldsymbol{x}_j^{\beta k})$ $(k = 1, 2, \ldots, K)$, γ_j^{α}, γ_j^{β}, and $\boldsymbol{x}_j^{\beta k}$ $(k = 0, 1, 2, \ldots, K)$ are unknown to adversary agent i. Parameters $\boldsymbol{x}_m^{\alpha k}, m \neq j$ and $\boldsymbol{\lambda}_{j,m}^{\alpha k}, m \neq j$ are known to adversary agent i only when agent m and agent i are neighbors. So the above system of $2DK$ equations contains at least $3DK + D + 2$ unknown variables, and adversary agent i cannot infer the gradient of local objective function ∇f_j by solving (5.11). Following the same line of argument, we can also obtain that an adversary agent i cannot solve a subset of equations in (5.11) to determine the gradient information either.

It is worth noting that after the optimization algorithm converges, adversary agent i can have another piece of information according to the KKT conditions [26]:

$$\nabla f_j(\boldsymbol{x}_j^*) = -\sum_{m\in\mathcal{N}_j} \boldsymbol{\lambda}_{j,m}^{\alpha *}. \tag{5.12}$$

If agent j's neighbors are also neighbors to the honest-but-curious agent i, the exact gradient of f_j at the optimal solution can be inferred by agent i. Therefore, agent j's gradient of local objective function ∇f_j will not be revealed to an honest-but-curious agent i at any point except the optimal solution. □

Corollary 6. *In Algorithm 4, agent j's gradient of local objective function ∇f_j at any point except the optimal solution will not be revealed to external eavesdroppers.*

Proof. The proof can be obtained following a similar line of reasoning of Theorem 10. External eavesdroppers can also establish the system of $2DK$ equations (5.11) to infer agent j's gradient ∇f_j. However, in this case, the number of unknowns, i.e., $\nabla f_j^{\alpha k}(\boldsymbol{x}_j^{\alpha k})$ $(k = 1, 2, \ldots, K)$, $\nabla f_j^{\beta k}(\boldsymbol{x}_j^{\beta k})$ $(k = 1, 2, \ldots, K)$, γ_j^{α}, γ_j^{β}, and $\boldsymbol{x}_j^{\beta k}$ $(k = 0, 1, 2, \ldots, K)$, adds up to $3DK + D + 2$, making the system of equations undetermined. Therefore, following the argument in the proof of Theorem 10, we can obtain that external eavesdroppers cannot infer the gradient of local objective function ∇f_j at any point except the optimal solution. □

Remark 14. *It is worth noting that if multiple adversary agents cooperate to infer the information of agent j, they can only establish a system of $2DK$ equations containing at least $3DK + D + 2$ unknown variables as well. Therefore, our algorithm can protect the privacy of agents against multiple honest-but-curious adversaries and external eavesdroppers.*

5.4 Convergence Analysis

In this section, we rigorously prove the convergence of Algorithm 4 under the following assumptions.

Assumption 6. *Function $\bar{f}(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{N} f_i(\tilde{\boldsymbol{x}}) : \mathbb{R}^D \to \mathbb{R}$ is strongly convex and continuously differentiable, i.e.,*

$$(\nabla \bar{f}(\tilde{\boldsymbol{x}}) - \nabla \bar{f}(\tilde{\boldsymbol{y}}))^T (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}}) \geq m_{\bar{f}} \parallel \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}} \parallel^2 .$$

Assumption 7. *Each local function $f_i : \mathbb{R}^D \to \mathbb{R}$ is convex and continuously differentiable.*

Assumption 8. *Each local function $f_i : \mathbb{R}^D \to \mathbb{R}$ has Lipschitz continuous gradients, i.e.,*

$$\parallel \nabla f_i(\tilde{\boldsymbol{x}}) - \nabla f_i(\tilde{\boldsymbol{y}}) \parallel \leq L_i \parallel \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}} \parallel .$$

Assumption 9. *$f_i^{\alpha k}$ is chosen under the following constraints:*

1) $f_i^{\alpha k}$ is convex and differentiable.

2) $f_i^{\beta k} = f_i - f_i^{\alpha k}$ is convex and differentiable.

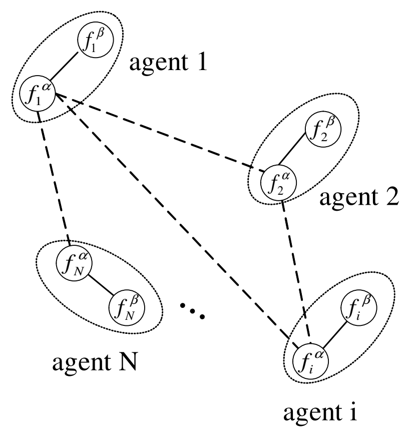
3) $f_i^{\alpha k}$ has Lipschitz continuous gradients, i.e. there exists an $L < +\infty$ such that

$$\parallel \nabla f_i^{\alpha k}(\tilde{\boldsymbol{x}}) - \nabla f_i^{\alpha k}(\tilde{\boldsymbol{y}}) \parallel \leq L \parallel \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}} \parallel .$$

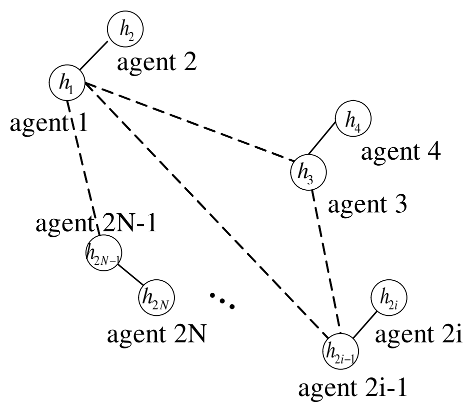
4) $f_i^{\beta k} = f_i - f_i^{\alpha k}$ has Lipschitz continuous gradients, i.e. there exists an $L < +\infty$ such that

$$\parallel \nabla f_i^{\beta k}(\tilde{\boldsymbol{x}}) - \nabla f_i^{\beta k}(\tilde{\boldsymbol{y}}) \parallel \leq L \parallel \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}} \parallel .$$

5) $\lim\limits_{k \to \infty} f_i^{\alpha k} \to f_i^{\alpha *}$ and $f_i^{\alpha k}(\tilde{\boldsymbol{x}})$ is bounded when $\tilde{\boldsymbol{x}}$ is bounded.



(a) $G = (V, E)$ of N agents



(b) Virtual network $G' = (V', E')$ of $2N$ agents

Figure 5.2: Function-decomposition based privacy-preserving decentralized optimization equals to converting the original network into a virtual network $G' = (V', E')$ of $2N$ agents.

It is worth noting that under Assumption 7 and Assumption 8, $f_i^{\alpha k}$ can be easily designed to meet Assumption 9. A quick example is $f_i^{\alpha k}(\tilde{\boldsymbol{x}}) = \boldsymbol{b}_i^{kT}\tilde{\boldsymbol{x}}$ where $\boldsymbol{b}_i^k \in \mathbb{R}^D$ is time-varying, and satisfies $\lim\limits_{k\to\infty} \boldsymbol{b}_i^k \to \boldsymbol{b}_i^*$ and $-\infty < \| \boldsymbol{b}_i^k \| < \infty$.

Because the function decomposition process amounts to converting the original network to a virtual network $G' = (V', E')$ of $2N$ agents, as shown in Fig. 5.2, we analyze the convergence of our algorithm based on the virtual network $G' = (V', E')$. To simplify and unify the notations, we relabel the local objective functions $f_i^{\alpha k}$ and $f_i^{\beta k}$ for all $i = 1, 2, \ldots, N$ as $h_1^k, h_2^k, \ldots, h_{2N}^k$. We relabel the associated states $\boldsymbol{x}_i^{\alpha k}$ and $\boldsymbol{x}_i^{\beta k}$ for all $i = 1, 2, \ldots, N$ as $\boldsymbol{x}_1^k, \boldsymbol{x}_2^k, \ldots, \boldsymbol{x}_{2N}^k$. In addition, we relabel parameters γ_i^α and γ_i^β for all $i = 1, 2, \ldots, N$ accordingly as $\gamma_1, \gamma_2, \ldots, \gamma_{2N}$. Then problem (5.4) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{x}_i \in \mathbb{R}^D,\, i \in \{1,2,\ldots,2N\}} \quad & \sum_{i=1}^{2N} h_i^k(\boldsymbol{x}_i) \\ \text{subject to} \quad & A\boldsymbol{x} = \mathbf{0} \end{aligned} \tag{5.13}$$

where $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, \ldots, \boldsymbol{x}_{2N}^T]^T \in \mathbb{R}^{2DN}$ and $A = [a_{m,l}] \otimes I_D \in \mathbb{R}^{D|E'| \times 2DN}$ is the edge-node incidence matrix of graph $\mathcal{G}'$ as defined in [118]. More specifically, $a_{m,l}$ is determined as

$$a_{m,l} = \begin{cases} 1 & \text{if the } m^{th} \text{ edge originates from agent } l, \\ -1 & \text{if the } m^{th} \text{ edge terminates at agent } l, \\ 0 & \text{otherwise.} \end{cases}$$

We define each edge $e_{i,j}$ originating from i and terminating at j and denote an edge as $e_{i,j} \in E'$ if $i < j$ is true or as $e_{j,i} \in E'$ otherwise.

Denote the iterating results in the kth step in Algorithm 4 as follows:

$$\begin{aligned}
\boldsymbol{x}^k &= [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, \ldots, \boldsymbol{x}_{2N}^{kT}]^T \in \mathbb{R}^{2DN}, \\
\boldsymbol{\lambda}^k &= [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E'} \in \mathbb{R}^{D|E'|}, \\
\boldsymbol{y}^k &= [\boldsymbol{x}^{kT}, \boldsymbol{\lambda}^{kT}]^T \in \mathbb{R}^{(|E'|+2N)D}
\end{aligned}$$

Further augment the coefficients γ_i $(i = 1, 2, \ldots, 2N)$ into the matrix form

$$U = \operatorname{diag}\{\gamma_1, \gamma_2, \ldots, \gamma_{2N}\} \otimes I_D \in \mathbb{R}^{2DN \times 2DN},$$

and N_i into the matrix form

$$\bar{D} = \operatorname{diag}\{N_1, N_2, \ldots, N_{2N}\} \otimes I_D \in \mathbb{R}^{2DN \times 2DN}.$$

Then we are in position to give the main results for this section:

Definition 2. *(Restricted strongly convex with respect to a point $\tilde{\boldsymbol{x}}^*$ [76]) A convex and differential function $f(\tilde{\boldsymbol{x}})$ is restricted strongly convex with respect to a point $\tilde{\boldsymbol{x}}^*$ if the following holds for all $\tilde{\boldsymbol{x}}$*

$$(\nabla f(\tilde{\boldsymbol{x}}) - \nabla f(\tilde{\boldsymbol{x}}^*))^T (\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^*) \geq m_f \parallel \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^* \parallel^2 \tag{5.14}$$

where $m_f > 0$ is a constant.

Lemma 1. *Define $r^k(\boldsymbol{x}) : \mathbb{R}^{2DN} \to \mathbb{R}$ as*

$$r^k(\boldsymbol{x}) = h^k(\boldsymbol{x}) + \frac{\rho(1-\tau)}{2} \parallel A\boldsymbol{x} \parallel^2$$

for $0 < \tau < 1$ where $h^k(\boldsymbol{x}) = \sum_{i=1}^{2N} h_i^k(\boldsymbol{x}_i)$, and $\bar{h}^k(\tilde{\boldsymbol{x}}) : \mathbb{R}^D \to \mathbb{R}$ is defined as

$$\bar{h}^k(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{2N} h_i^k(\tilde{\boldsymbol{x}}).$$

Let $\boldsymbol{x}^*$ be the optimal solution to (5.13). *If Assumptions 6, 7, 8, and 9 are satisfied, we have that $r^k(\boldsymbol{x})$ is* *restricted strongly convex with respect to the optimal solution $\boldsymbol{x}^*$, i.e., the following holds for all $\boldsymbol{x}$*

$$(\nabla r(\boldsymbol{x}) - \nabla r(\boldsymbol{x}^*))^T(\boldsymbol{x} - \boldsymbol{x}^*) \geq m_r \parallel \boldsymbol{x} - \boldsymbol{x}^* \parallel^2 \tag{5.15}$$

where

$$m_r \geq \{\frac{m_{\bar{f}}}{2N} - 2L\upsilon, \frac{A_{\min}\rho(1-\tau)}{1+\frac{1}{\upsilon^2}}\}, \tag{5.16}$$

for any $\upsilon \in (0, \frac{m_{\bar{f}}}{4NL})$ with $A_{\min}$ the smallest nonzero eigenvalue of A^TA, $m_{\bar{f}}$ given in Assumption 6, and L given in Assumption 9.

Proof. Because $\bar{h}^k(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{2N} h_i^k(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{N} f_i(\tilde{\boldsymbol{x}}) = \bar{f}(\tilde{\boldsymbol{x}})$ is strongly convex, and the matrix A here is the same as the matrix E_o in [76], according to Lemma 1 in [76] and Appendix 1 in [102], we have the Lemma. □

Lemma 2. *Let $\boldsymbol{x}^*$ be the optimal solution to* (5.13), *$\boldsymbol{\lambda}^{k*}$ be the optimal multiplier to* (5.13) *at iteration k, and $\boldsymbol{y}^{k*}$ be the augmented vector $[\boldsymbol{x}^{*T}, \boldsymbol{\lambda}^{k*T}]^T$. Further define $Q = U + \bar{D} - A^TA$, $H = \mathrm{diag}\{\rho Q, \frac{1}{\rho}I_{D|E'|}\}$, $A_{\max}$ and $A_{\min}$ as the respective maximal and minimal nonzero eigenvalues of A^TA. Then we have*

$$\parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H \leq \frac{\parallel \boldsymbol{y}^k - \boldsymbol{y}^{k+1*} \parallel_H}{\sqrt{1+\delta}} \tag{5.17}$$

if $U + \bar{D} - A^TA$ is positive semi-definite and Assumptions 6, 7, 8, and 9 are satisfied. In (5.17), *$\parallel \tilde{\boldsymbol{x}} \parallel_H = \sqrt{\tilde{\boldsymbol{x}}^T H \tilde{\boldsymbol{x}}}$ and*

$$\delta = \min\{\frac{(u-1)\tau A_{\min}}{2uQ_{\max}}, \frac{2m_r\rho\tau(u-1)A_{\min}}{\phi}\} \tag{5.18}$$

where $u > 1$ is an arbitrary constant, $Q_{\max}$ is the largest eigenvalue of Q, m_r, L are given in Assumptions 6 and 9, respectively, and

$$\phi = u(u-1)L^2 + \rho^2\tau A_{\min}Q_{\max}(u-1) + 2\rho^2(1-\tau)^2 u A_{\max}^2.$$

Proof. The results can be obtained following a similar line of reasoning in [65]. The detailed proof is given in the Appendix B. □

Lemma 3. *Let $\boldsymbol{x}^*$ be the optimal solution to* (5.13), *$\boldsymbol{\lambda}^{k*}$ be the optimal multiplier to* (5.13) *at iteration k, and $\boldsymbol{y}^{k*}$ be the augmented vector $[\boldsymbol{x}^{*T}, \boldsymbol{\lambda}^{k*T}]^T$. Further define $Q = U + \bar{D} - A^TA$ and $H = \mathrm{diag}\{\rho Q, \frac{1}{\rho}I_{D|E'|}\}$. Then we have*

$$\parallel \boldsymbol{y}^k - \boldsymbol{y}^{k+1*} \parallel_H \leq \parallel \boldsymbol{y}^k - \boldsymbol{y}^{k*} \parallel_H + p(k) \tag{5.19}$$

if $U + \bar{D} - A^T A$ is positive semi-definite and Assumptions 6, 7, 8, and 9 are satisfied. In (5.19),

$$p(k) = \frac{1}{\sqrt{\rho \tau A_{\min}}} \parallel \nabla h^{k+1}(\boldsymbol{x}^*) - \nabla h^k(\boldsymbol{x}^*) \parallel \tag{5.20}$$

where $h^k(\boldsymbol{x}) = \sum_{i=1}^{2N} h_i^k(\boldsymbol{x}_i)$.

Proof. The results can be obtained following a similar line of reasoning in [65]. The detailed proof is given in the Appendix B. □

Lemma 4. *Let $\boldsymbol{x}^*$ be the optimal solution to* (5.13)*, $\boldsymbol{\lambda}^{k*}$ be the optimal multiplier to* (5.13) *at iteration k, and $\boldsymbol{y}^{k*}$ be the augmented vector $[\boldsymbol{x}^{*T}, \boldsymbol{\lambda}^{k*T}]^T$. Further define $Q = U + \bar{D} - A^T A$ and $H = \operatorname{diag}\{\rho Q, \frac{1}{\rho} I_{D|E'|}\}$. Then we have*

$$\parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H \leq \frac{\parallel \boldsymbol{y}^k - \boldsymbol{y}^{k*} \parallel_H}{\sqrt{1+\delta}} + \frac{p(k)}{\sqrt{1+\delta}} \tag{5.21}$$

if $U + \bar{D} - A^T A$ is positive semi-definite and Assumptions 6, 7, 8, and 9 are satisfied.

Proof. Combining (5.17) and (5.19), we obtain the result directly. □

Lemma 4 indicates that $\parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H$ converges linearly to a neighborhood of 0. When the local objective function is not dynamically changing, i.e. $p(k) = 0$, we have that the proximal Jacobian ADMM has a Q-linear convergence rate without requiring all local objective functions to be strongly convex, which gives us the following theorem.

Theorem 11. *Algorithm 4 is guaranteed to converge to the optimal solution to* (5.13) *with a Q-linear convergence rate if $U + \bar{D} - A^T A$ is positive semi-definite, Assumptions 6, 7, 8, and 9 are satisfied, and $f_i^{\alpha k}$ is time-invariant.*

Proof. When $f_i^{\alpha k}$ is time-invariant, we have $p(k) = 0$. Then from Lemma 4, we have the theorem. □

Remark 15. *[76] also achieves Q-linear convergence rate for ADMM under dummy variables. However, the introduced dummy variables increase the requirement on computational and memory resources. Furthermore, different from [52] which establishes R-linear[2] convergence rate under a sufficiently small dual step-size, our approach achieves a faster Q-linear convergence rate without such a constraint.*

When $f_i^{\alpha k}$ is time-variant, we have the following theorem.

[2]For a sequence $\{\boldsymbol{x}^k\}$ converging to $\boldsymbol{x}^*$ in some norm, R-linear convergence rate is achieved if there exists a $\lambda \in (0, 1)$ and some positive constant C such that $\parallel \boldsymbol{x}^k - \boldsymbol{x}^* \parallel \leq C\lambda^k$ holds for all k [85].

Theorem 12. *Algorithm 4 is guaranteed to converge to the optimal solution to* (5.13) *if $U + \bar{D} - A^T A$ is positive semi-definite and Assumptions 6, 7, 8, and 9 are satisfied.*

Proof. The proof is provided in the Appendix B. □

5.5 Numerical Simulations

In this section, we first illustrate the effectiveness of the proposed approach. Then we compare our approach with the differential-privacy based algorithm in [54] and the encryption based algorithm in Chapter 3. We conducted numerical experiments on the following global objective function

$$\tilde{f}(\tilde{\boldsymbol{x}}) = \sum_{i=1}^{N} \frac{1}{2} \parallel H_i \tilde{\boldsymbol{x}} - \boldsymbol{y}_i \parallel^2, \tag{5.22}$$

which makes the optimization problem (3.1) become

$$\min_{\tilde{\boldsymbol{x}}} \qquad \sum_{i=1}^{N} \frac{1}{2} \parallel H_i \tilde{\boldsymbol{x}} - \boldsymbol{y}_i \parallel^2 \tag{5.23}$$

with $\boldsymbol{y}_i \in \mathbb{R}^D$ and $H_i \in \mathbb{R}^{D \times D}$ a diagonal matrix. Hence, each agent i deals with a private local objective function

$$f_i(\boldsymbol{x}_i) = \frac{1}{2} \parallel H_i \boldsymbol{x}_i - \boldsymbol{y}_i \parallel^2, \forall i \in \{1, 2, \ldots, N\}. \tag{5.24}$$

We used the above function (5.22) because it is easy to verify whether the obtained solution is the minimal value of the original optimization problem, which is $(\sum_{i=1}^{N} H_i^T H_i)^{-1} (\sum_{i=1}^{N} H_i \boldsymbol{y}_i)$. Furthermore, (5.22) makes it easy to compare with [54], whose verification is also based on (5.22).

5.5.1 Evaluation of Our Approach

To solve the optimization problem (5.22), $f_i^{\alpha k}(\tilde{\boldsymbol{x}})$ was set to $f_i^{\alpha k}(\tilde{\boldsymbol{x}}) = (\boldsymbol{b}_i^k)^T \tilde{\boldsymbol{x}}$ in all simulations, where $\boldsymbol{b}_i^k$ was set to $\boldsymbol{b}_i^k = \frac{1}{k+1}\boldsymbol{c}_i + \boldsymbol{d}_i$ with $\boldsymbol{c}_i \in \mathbb{R}^D$ and $\boldsymbol{d}_i \in \mathbb{R}^D$ being constants private to agent i. Fig. 5.4 visualizes the evolution of $\boldsymbol{x}_{i1}^{\alpha}$ and $\boldsymbol{x}_{i1}^{\beta}$ $(i = 1, 2, ..., 6)$ in one specific run where $D = 2$ and the network deployment is illustrated in Fig. 5.3. Fig. 5.5 visualizes the evolution of $\boldsymbol{x}_{i2}^{\alpha}$ and $\boldsymbol{x}_{i2}^{\beta}$ $(i = 1, 2, ..., 6)$. Here, $\boldsymbol{x}_{ij}^{\alpha}$ denotes the jth element of $\boldsymbol{x}_i^{\alpha}$ and $\boldsymbol{x}_{ij}^{\beta}$ denotes the jth element of $\boldsymbol{x}_i^{\beta}$. All $\boldsymbol{x}_i^{\alpha}$ and $\boldsymbol{x}_i^{\beta}$ $(i = 1, 2, ..., 6)$

converged to the optimal solution.

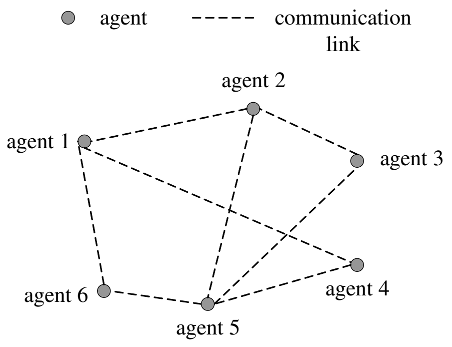


Figure 5.3: A network of six agents ($N = 6$).

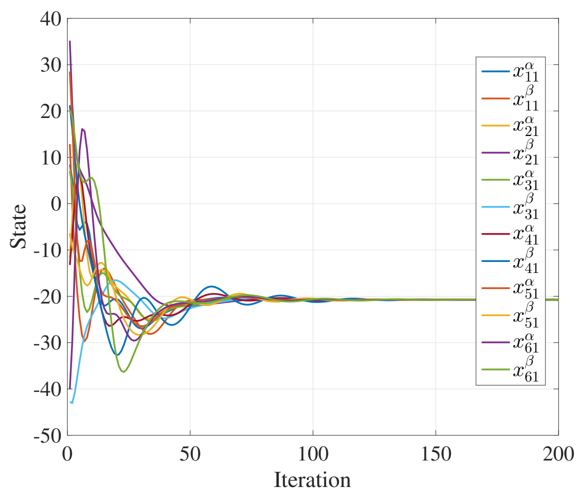


Figure 5.4: The evolution of $\boldsymbol{x}_{i1}^{\alpha}$ and $\boldsymbol{x}_{i1}^{\beta}$ $(i = 1, 2, ..., 6)$ in one specific run.

5.5.2 Comparison with the algorithm in [54]

Under the network deployment in Fig. 5.3, we compared our privacy-preserving approach with the differential-privacy based algorithm in [54]. We simulated the algorithm in [54] under seven different privacy levels:

$$\epsilon = 0.2, 1, 10, 20, 30, 50, 100.$$

In the objective function (5.22), H_i was set to the identity matrix and $\boldsymbol{y}_i$ was set to $\boldsymbol{y}_i = [0.1\times(i-1)+0.1; 0.1\times(i-1)+0.2]$. The domain of optimization for the algorithm in [54] was set to $\mathcal{X} = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}$.

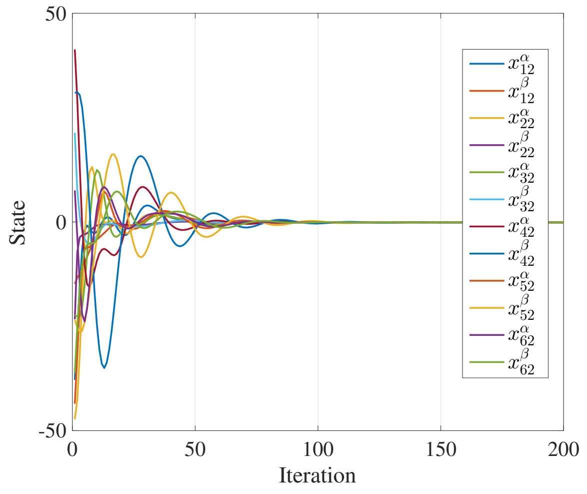


Figure 5.5: The evolution of $\boldsymbol{x}_{i2}^{\alpha}$ and $\boldsymbol{x}_{i2}^{\beta}$ $(i = 1, 2, ..., 6)$ in one specific run.

Note that the optimal solution $[0.35; 0.45]$ resided in $\mathcal{X}$. Detailed parameter settings for the algorithm in [54] were given as $n = 2$, $c = 0.5$, $q = 0.8$, $p = 0.9$, and

$$a_{ij} = \begin{cases} 0.2 & j \in \mathcal{N}_i, \\ 0 & j \notin \mathcal{N}_i, j \neq i, \\ 1 - \sum\limits_{j \in \mathcal{N}_i} a_{ij} & i = j, \end{cases} \tag{5.25}$$

for $i = 1, 2, ..., 6$. In addition, the performance index d in [54] was used to quantify the optimization error here, which was computed as the average value of squared distances with respect to the optimal solution over M runs [54], i.e.,

$$d = \frac{\sum\limits_{i=1}^{6} \sum\limits_{l=1}^{M} \parallel \boldsymbol{x}_i^l - [0.35; 0.45] \parallel^2}{6M}.$$

Here $\boldsymbol{x}_i^l$ is the obtained solution of agent i in the lth run. For our approach, $\boldsymbol{x}_i^l$ was calculated as the average of $\boldsymbol{x}_i^{\alpha l}$ and $\boldsymbol{x}_i^{\beta l}$.

Simulation results from 5,000 runs showed that our approach converged to $[0.35; 0.45]$ with an error $d = 6.5 \times 10^{-6}$, which is negligible compared with the simulation results under the algorithm in [54] (cf. Fig. 5.6, where each differential privacy level was implemented for 5,000 times). The results confirm the trade-off between privacy and accuracy in differential-privacy based approaches.

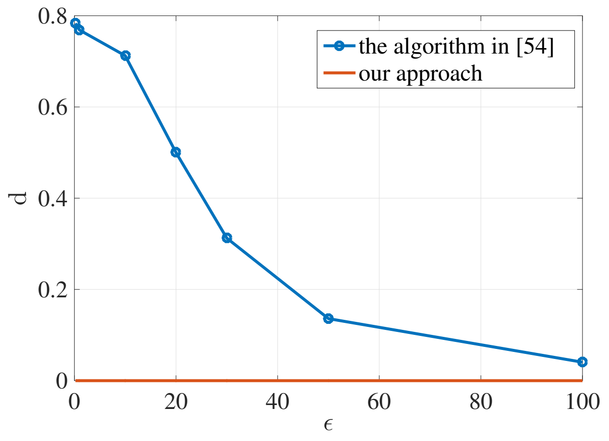


Figure 5.6: The comparison of our approach with the algorithm in [54] in terms of optimization error.

5.5.3 Comparison with the algorithm in Chapter 3

We also compared our approach with the privacy-preserving optimization algorithm in Chapter 3, which is based on ADMM and partially homomorphic encryption. The network communication topology used for comparison is a ring network of 30 agents and the global objective function used is (5.22) with H_i $(i = 1, 2, .., 6)$ set to identity matrix and $\boldsymbol{y}_i \in \mathbb{R}^2$. The initial states were set to the same values for both algorithms ($\boldsymbol{x}_i^{\alpha 0} = \boldsymbol{x}_i^{\beta 0} = \boldsymbol{x}_i^0$). Fig. 5.7 visualizes the evolution of $\boldsymbol{x}_i^{\alpha}$ and $\boldsymbol{x}_i^{\beta}$ in our approach whereas Fig. 5.8 visualizes the evolution of $\boldsymbol{x}_i$ of the algorithm in Chapter 3 when $\bar{b}$ in Chapter 3 was set to 1.5. It can be seen that our encryption-free approach has comparable convergence rate with the partially homomorphic encryption based algorithm in Chapter 3.

5.6 Summaries

In this chapter, we proposed a novel approach to enabling privacy-preservation in decentralized optimization based on function decomposition, which neither compromises the optimality of optimization nor relies on an aggregator or third party. Theoretical analysis confirms that an honest-but-curious adversary cannot infer the information of neighboring agents even by recording and analyzing the information exchanged in multiple iterations. In addition, our approach can also avoid an external eavesdropper from inferring the information of participating agents. Compared with encryption-based approaches which suffer from heavy computational and communication burden, the proposed approach incurs little extra computational

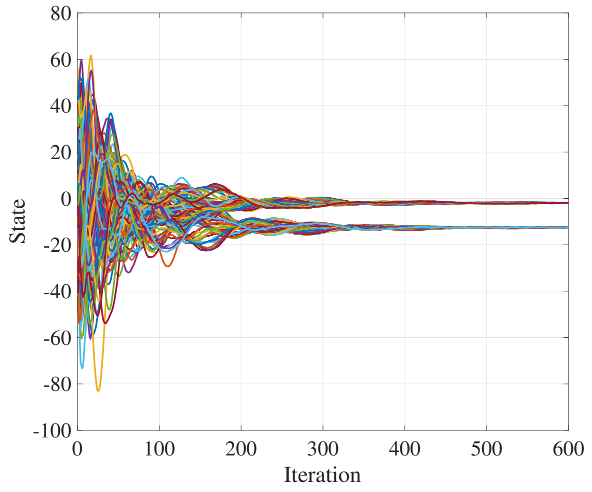


Figure 5.7: The evolution of $\boldsymbol{x}_{i1}^{\alpha}$ and $\boldsymbol{x}_{i1}^{\beta}$ $(i = 1, 2, ..., 6)$ in our approach.

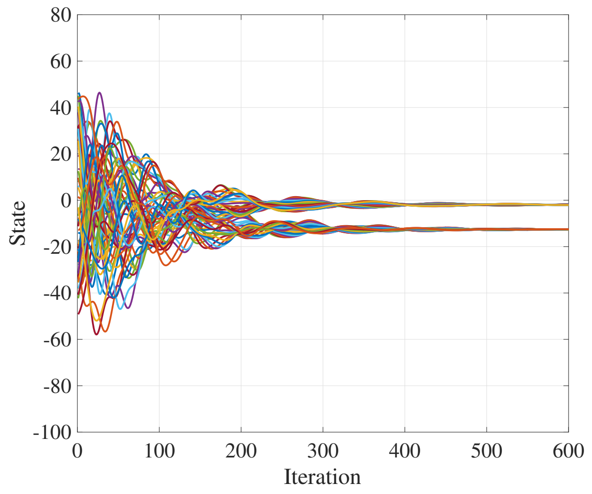


Figure 5.8: The evolution of $\boldsymbol{x}_i$ of the algorithm in Chapter 3.

and communication overhead. Furthermore, we prove that when the global objective function is strongly convex, proximal Jacobian ADMM can achieve Q-linear convergence rate even when local individual objective functions are only convex, which generalizes existing proximal Jacobian ADMM results requiring local objective functions to be strongly convex to achieve Q-linear convergence rate. Numerical simulation results confirmed the effectiveness of the proposed approach.

Chapter 6

Decentralized Non-convex Event Localization via ADMM

Event localization plays a fundamental role in many wireless sensor network applications such as environmental monitoring, homeland security, medical treatment, and health care, and it is essentially a non-convex and non-smooth problem. In this chapter, we address such a problem in a completely decentralized way based on augmented Lagrangian methods and alternating direction method of multipliers (ADMM). The main contributions of this chapter are as follows:

1. An algorithm is applied to directly solve the general non-smooth and non-convex event localization problem without using convex relaxation. The avoidance of convex relaxation is significant in that convex relaxation based methods generally suffer from high computational complexity. It is worth noting that recently results have emerged for ADMM in non-convex optimization [53, 72]. However, [53] requires objective functions to have Lipschitz continuous derivatives and [72] requires objective functions to be continuously differentiable and have bounded gradient, neither of which can be satisfied by the non-smooth event localization problem considered in this chapter. Furthermore, the non-convex and non-smooth optimization approach in [116] is not applicable to our problem either because it requires some parts of the objective function to be restricted prox-regular[1], which is not the case here;

[1]Restricted prox-regularity is defined in Definition 2 of [116]: For a lower semi-continuous function f, let $M \in \mathbb{R}_+, f : \mathbb{R}^D \to \mathbb{R} \cup \{\infty\}$, and define the exclusion set $S_M := \{\boldsymbol{x} \in \text{dom}(f) : \| \boldsymbol{d} \| > M, \forall \boldsymbol{d} \in \partial f(\boldsymbol{x})\}$. f is called restricted prox-regular if, for any $M > 0$ and bounded set $T \subseteq \text{dom} f$, there exists $\gamma > 0$ such that $f(\boldsymbol{y}) + \frac{\gamma}{2} \| \boldsymbol{x} - \boldsymbol{y} \|^2 \geq f(\boldsymbol{x}) + \langle \boldsymbol{d}, \boldsymbol{y} - \boldsymbol{x} \rangle, \forall \boldsymbol{x} \in T \backslash S_M, \boldsymbol{y} \in T, \boldsymbol{d} \in \partial f(\boldsymbol{x}), \| \boldsymbol{d} \| \leq M$.

2. The proposed algorithm takes full advantages of alternating direction method of multipliers which decomposes a general optimization problem into multiple local optimization subproblems with each subproblem solved by an individual sensor. Through cooperations in the computation process among neighboring sensors, a consistent estimate of the event position across the entire network can be achieved. Therefore, compared with centralized approach in which a processing center performs the whole heavy computation, the algorithm is highly scalable, flexible, robust to network topology changes, and thus is more favorable in practical implement;

3. Numerical simulations show that the proposed algorithm achieves better localization accuracy than existing distributed projection-based approaches when the target is within the convex hull of localization sensors. When the target is outside the convex hull, numerical simulations show that the proposed approach has a higher probability to converge to the target event location than existing projection-based approaches.

The rest of this chapter is organized as follows: Sec. 6.1 states the formulation of the problem. To solve the problem, a decentralized algorithm is proposed in Sec. 6.2 and its convergence properties are analyzed in Sec. 6.3. Sec. 6.4 gives numerical simulation results of the algorithm and its comparison with existing results. In the end, a conclusion is made in Sec. 6.5.

6.1 Problem Statement

We suppose that the sensor network for event localization is composed of N sensors and the position of sensor i is denoted as $\boldsymbol{a}_i \in \mathbb{R}^D$, where D ($D \in \{1, 2, 3\}$) is the dimension. If there exists a communication link between sensors i and j, we say that sensors i and j can communicate and exchange information. We further denote the unknown position of the target event as $\boldsymbol{x} \in \mathbb{R}^D$. Then the noisy range r_i between sensor i and the target event is denoted as:

$$r_i = d_i + v_i,$$

where $d_i = \| \boldsymbol{x} - \boldsymbol{a}_i \|$ denotes the real distance between sensor i and the target event, and v_i denotes the measurement noise.

Suppose that both $\boldsymbol{a}_i$ and r_i are available to sensor i and are private (i.e., $\boldsymbol{a}_i$ and r_i are only known to sensor i). Furthermore, we assume that the sensor network is connected, i.e., there exists a (multi-hop) path between every pair of sensors in the network. The event localization problem addressed in this chapter is to estimate the unknown event position $\boldsymbol{x}$ from available sensor positions $\boldsymbol{a}_i$ and noisy range measurements r_i ($i = 1, 2, \ldots, N$) with respect to the target event.

Remark 16. *The setting of not exchanging sensors' positions makes great sense in practical applications. For example, on the battlefield where soldiers wear sensor devices to locate gunfire, exchanging soldiers' positions takes great risks of being intercepted to opponents or compromised teammate, which will put soldiers in danger [6].*

Assume that $\boldsymbol{v} = (v_1, v_2, ..., v_N)^T$ follows a standard Gaussian distribution and its covariance matrix is a diagonal matrix with equal diagonal elements, then the position estimate of the target event $\boldsymbol{x}$ is the solution to the following maximum likelihood problem [11]:

$$\min_{\boldsymbol{x}} \quad \sum_{i=1}^{N} f_i(\boldsymbol{x}), \tag{6.1}$$

where

$$f_i(\boldsymbol{x}) = \sum_{i=1}^{N} (\| \boldsymbol{x} - \boldsymbol{a}_i \| - r_i)^2. \tag{6.2}$$

$f_i(\boldsymbol{x})$ in (6.2) is non-convex, non-smooth and does not satisfy the restricted prox-regular condition, which invalidates the application of existing results [53,72,116]. In this chapter, we will show that by designing the penalty parameter, we can address this non-convex and non-smooth problem (6.1) via ADMM directly without using convex relaxation. We first propose the algorithm in the following section and then analyze its convergence in Sec. 6.3.

6.2 Proposed Decentralized Algorithm

6.2.1 Problem Reformulation

To address problem (6.1) in a decentralized way via ADMM, first we denote the communication pattern of the sensor network as an undirected graph $G = \{V, E\}$ [14], where V is the set of N sensors and E is the set of undirected edges (communication links) among the sensors. We assume that the undirected network is connected. Let $|E|$ be the total number of undirected edges. Then if there is an edge between sensors i and j, we denote it as $e_{i,j} \in E$ and say that sensor j is a neighboring sensor of i (sensor i is a neighboring sensor of j as well). Note that here both $e_{i,j}$ and $e_{j,i}$ denote the same edge, so we only use the expression of $e_{i,j}$ (if $i < j$) or $e_{j,i}$ (if $j < i$) to avoid repetition. We denote $\mathcal{N}_i$ as the neighboring set of sensor i and N_i as the number of neighboring sensors in $\mathcal{N}_i$. Assume that each sensor has an estimate of the target event position $\boldsymbol{x}$, denoted as $\boldsymbol{x}_i$. Then each sensor is associated with a local cost function $f_i(\boldsymbol{x}_i)$ and all local cost functions are combined into the general problem in (6.1). We also suppose that the local cost function f_i is only known to sensor i since $\boldsymbol{a}_i$ and r_i are private and only available to sensor i. Therefore, to reach consistency on the estimated target event position across the entire network, we need to impose the constraints $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ and $\boldsymbol{x}_j = \boldsymbol{z}_{i,j}$ if there exists an edge $e_{i,j}$ between sensors i and j. Here, $\boldsymbol{z}_{i,j}(i < j)$ is an auxiliary item. We use the constraint $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ instead of $\boldsymbol{x}_i = \boldsymbol{z}$ in [16] because the constraint $\boldsymbol{x}_i = \boldsymbol{z}$ requires a central node to collect all $\boldsymbol{x}_i$ for $i = 1, 2, ..., N$ to update $\boldsymbol{z}$ whereas under constraint $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ individual nodes can update $\boldsymbol{z}_{i,j}$ in a decentralized way.

Now problem (6.1) can be reformulated as a distributed ADMM described as follows:

$$\begin{aligned} \min_{\boldsymbol{x}_i,\, i \in \{1,2,\ldots,N\}, \boldsymbol{z}_{i,j}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) \\ \text{subject to} \quad & \boldsymbol{x}_i = \boldsymbol{z}_{i,j}, \quad \boldsymbol{x}_j = \boldsymbol{z}_{i,j}, \quad \forall e_{i,j} \in E, \end{aligned} \tag{6.3}$$

or in a more compact form:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & C\boldsymbol{x} + F\boldsymbol{z} = 0, \end{aligned} \tag{6.4}$$

where $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_N^T]^T$, $g(\boldsymbol{z}) = 0$ is the identical zero function, $\boldsymbol{z} = [\boldsymbol{z}_{i,j}]_{ij, e_{i,j} \in E}$, matrices C and F are defined similarly to [103], which we recapitulate here: $C = [C_1; C_2]$; both $C_1 \in \mathbb{R}^{|E|D \times ND}$ and $C_2 \in \mathbb{R}^{|E|D \times ND}$ consist of $|E| \times N$ blocks of $D \times D$ matrices. If there exists an edge $e_{i,j}$ between sensors i and j, and $\boldsymbol{z}_{i,j}$ is the qth block in $\boldsymbol{z}$, then the (q, i)th block in C_1 and (q, j)th block in C_2 are identity matrices I_D for all q, i, and j. The other blocks in C_1 and C_2 are identical zero matrices O_D. Matrix $F = [-I_{|E|D}; -I_{|E|D}]$ with $I_{|E|D}$ being an $|E|D \times |E|D$ identity matrix. It is worth noting that the constraints $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ and $\boldsymbol{x}_j = \boldsymbol{z}_{i,j}$ for all $e_{i,j} \in E$ imply that the feasible set of (6.4) is $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^D | A\boldsymbol{x} = \mathbf{0}\}$, where $A = [a_{m,n}] \otimes I_D \in \mathbb{R}^{|E|D \times ND}$ is the edge-node incidence matrix of graph G as defined in [118]. The symbol $\otimes$ denotes Kronecker product. The $a_{m,n}$ element is defined as

$$a_{m,n} = \begin{cases} 1 & \text{if the } m^{th} \text{ edge originates from agent } n, \\ -1 & \text{if the } m^{th} \text{ edge terminates at agent } n, \\ 0 & \text{otherwise.} \end{cases}$$

Here we define that each edge $e_{i,j}$ originates from agent i and terminates at agent j.

In this reformulation, the imposed constraints $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}, \boldsymbol{x}_j = \boldsymbol{z}_{i,j}, \forall e_{i,j} \in E$ in (6.3) require neighboring sensors to exchange copies of local estimated event positions. Through an exchange of intermediate computational results, it is guaranteed that a consistency of individual local estimates of event positions $\boldsymbol{x}_i$ across the entire network can be achieved. Now we are in place to solve (6.3).

6.2.2 Proposed Algorithm

Let $\boldsymbol{\lambda}^i_{i,j}$ be the Lagrange multiplier relevant to the constraint $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ and $\boldsymbol{\lambda}^j_{i,j}$ be the Lagrange multiplier relevant to the constraint $\boldsymbol{x}_j = \boldsymbol{z}_{i,j}$. Parameter ρ_i is the penalty parameter associated with sensor i and $\rho_i > 0$. Then we can denote the regularized augmented Lagrangian function of problem (6.3) as

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} f_i(\boldsymbol{x}_i) + \sum_{e_{i,j} \in E} (\boldsymbol{\lambda}^{iT}_{i,j}(\boldsymbol{x}_i - \boldsymbol{z}_{i,j}) + \boldsymbol{\lambda}^{jT}_{i,j}(\boldsymbol{x}_j - \boldsymbol{z}_{i,j})) \\ + \sum_{e_{i,j} \in E} (\frac{\rho_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{z}_{i,j} \parallel^2 + \frac{\rho_j}{2} \parallel \boldsymbol{x}_j - \boldsymbol{z}_{i,j} \parallel^2), \end{aligned} \tag{6.5}$$

where $\boldsymbol{\lambda}$ is the shorthand notion for $\boldsymbol{\lambda}^i_{i,j}$ and $\boldsymbol{\lambda}^j_{i,j}$, $\boldsymbol{\rho} = \text{diag}\{\rho_i \mathbf{1}_{DN_i}\}_{i=\{1,2,\ldots,N\}}$, $\mathbf{1}_{DN_i}$ is a column vector of length DN_i and all its entries are one. Applying ADMM, we can get the following three recursions:

$$\boldsymbol{x}^{t+1} \in \text{argmin}_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\rho}^t}(\boldsymbol{x}, \boldsymbol{z}^t, \boldsymbol{\lambda}^t), \tag{6.6}$$

$$\boldsymbol{z}^{t+1} = \text{argmin}_{\boldsymbol{z}} \mathcal{L}_{\boldsymbol{\rho}^t}(\boldsymbol{x}^{t+1}, \boldsymbol{z}, \boldsymbol{\lambda}^t), \tag{6.7}$$

$$\boldsymbol{\lambda}^{t+1} = \boldsymbol{\lambda}^t + \boldsymbol{\rho}^t(C\boldsymbol{x}^{t+1} + F\boldsymbol{z}^{t+1}). \tag{6.8}$$

The function $\mathcal{L}_{\boldsymbol{\rho}^t}(\boldsymbol{x}, \boldsymbol{z}^t, \boldsymbol{\lambda}^t)$ in (6.6) may have more than one local minimum, so we use $\boldsymbol{x}^{t+1} \in \text{argmin}_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\rho}^t}(\boldsymbol{x}, \boldsymbol{z}^t, \boldsymbol{\lambda}^t)$ instead of $\boldsymbol{x}^{t+1} = \text{argmin}_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\rho}^t}(\boldsymbol{x}, \boldsymbol{z}^t, \boldsymbol{\lambda}^t)$. Here "argmin" is used to indicate finding local minima. The above recursions from (6.6) to (6.8) can be realized in a decentralized way, which is described in Algorithm 5.

Algorithm 5

Initial Setup: Each sensor initializes $\boldsymbol{x}_i^0$, $\boldsymbol{\lambda}_{i,j}^{i0}$, $\boldsymbol{\lambda}_{i,j}^{j0}$, ρ_i^0, and exchanges $\boldsymbol{x}_i^0$ with neighboring sensors. Then it sets $\boldsymbol{z}_{i,j}^0 = \frac{\boldsymbol{x}_i^0+\boldsymbol{x}_j^0}{2}$.
Input: $\boldsymbol{x}_i^t$, $\boldsymbol{z}_{i,j}^t$, $\boldsymbol{\lambda}_{i,j}^{it}$, $\boldsymbol{\lambda}_{i,j}^{jt}$, ρ_i^t
Output: $\boldsymbol{x}_i^{t+1}$, $\boldsymbol{z}_{i,j}^{t+1}$, $\boldsymbol{\lambda}_{i,j}^{i(t+1)}$, $\boldsymbol{\lambda}_{i,j}^{j(t+1)}$, ρ_i^{t+1}

1. Each sensor updates its local vector $\boldsymbol{x}_i^{t+1}$:

$$\boldsymbol{x}_i^{t+1} \in \text{argmin}_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i, i<j} (\boldsymbol{\lambda}_{i,j}^{itT}\boldsymbol{x}_i + \frac{\rho_i^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{z}_{i,j}^t \parallel^2) + \sum_{j\in\mathcal{N}_i, i>j} (\boldsymbol{\lambda}_{j,i}^{itT}\boldsymbol{x}_i + \frac{\rho_i^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{z}_{j,i}^t \parallel^2).$$

   To simplify and unify the above expression, we introduce three notions: we use $\boldsymbol{\lambda}_{i,j}^i$ to represent $\boldsymbol{\lambda}_{j,i}^i$, $\boldsymbol{\lambda}_{i,j}^j$ to represent $\boldsymbol{\lambda}_{j,i}^j$, and $\boldsymbol{z}_{i,j}$ to represent $\boldsymbol{z}_{j,i}$ in each sensor i. Then the above equation can be rewritten as follows:

$$\boldsymbol{x}_i^{t+1} \in \text{argmin}_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i} (\boldsymbol{\lambda}_{i,j}^{itT}\boldsymbol{x}_i + \frac{\rho_i^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{z}_{i,j}^t \parallel^2). \tag{6.9}$$

2. Each sensor sends its local vector $\boldsymbol{x}_i^{t+1}$ and ρ_i^t to neighboring sensors;

3. Each sensor computes $\boldsymbol{z}_{i,j}^{t+1}$ for $j \in \mathcal{N}_i$:

$$\boldsymbol{z}_{i,j}^{t+1} = \text{argmin}_{\boldsymbol{z}_{i,j}} - (\boldsymbol{\lambda}_{i,j}^{itT} + \boldsymbol{\lambda}_{i,j}^{jtT})\boldsymbol{z}_{i,j} + (\frac{\rho_i^t}{2} \parallel \boldsymbol{x}_i^{t+1} - \boldsymbol{z}_{i,j} \parallel^2 + \frac{\rho_j^t}{2} \parallel \boldsymbol{x}_j^{t+1} - \boldsymbol{z}_{i,j} \parallel^2). \tag{6.10}$$

   Note that here the values of $\boldsymbol{z}_{i,j}^{t+1}$ computed by sensors i and j are the same. This problem is easy to solve, whose approximate solution is:

$$\boldsymbol{z}_{i,j}^{t+1} = \frac{1}{\rho_i^t + \rho_j^t}(\rho_i^t\boldsymbol{x}_i^{t+1} + \rho_j^t\boldsymbol{x}_j^{t+1} + \boldsymbol{\lambda}_{i,j}^{it} + \boldsymbol{\lambda}_{i,j}^{jt}); \tag{6.11}$$

4. Each sensor computes for $j \in \mathcal{N}_i$:

$$\boldsymbol{\lambda}_{i,j}^{i(t+1)} = \boldsymbol{\lambda}_{i,j}^{it} + \rho_i^t(\boldsymbol{x}_i^{t+1} - \boldsymbol{z}_{i,j}^{t+1}), \tag{6.12}$$
$$\boldsymbol{\lambda}_{i,j}^{j(t+1)} = \boldsymbol{\lambda}_{i,j}^{jt} + \rho_j^t(\boldsymbol{x}_j^{t+1} - \boldsymbol{z}_{i,j}^{t+1}). \tag{6.13}$$

   Note that here the values of $\boldsymbol{\lambda}_{i,j}^{i(t+1)}$ and $\boldsymbol{\lambda}_{i,j}^{j(t+1)}$ computed by sensors i and j are the same;

5. Each sensor updates $\rho_i^t \to \rho_i^{t+1}$. Detailed updating rules are given in Sec. 6.3;

6. Set $t = t + 1$, and go to 1.

Now our focus is to find a closed-form solution to subproblem (6.9). First, let us introduce a Lemma:

Lemma 5. *Let $\mathcal{A} = \{\boldsymbol{a}_1, ..., \boldsymbol{a}_N\}$ be the set of N sensors' positions and*

$$L_i(\boldsymbol{x}_i) = f_i(\boldsymbol{x}_i) + \sum_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{i,j}^{itT} \boldsymbol{x}_i + \frac{\rho_i^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{z}_{i,j}^t \parallel^2). \tag{6.14}$$

Then for any $i = 1, 2, ..., N$, the following statements are true:

Statement (1): Every $\bar{\boldsymbol{a}} \in \mathcal{A}$ is not a local minimum of problem (6.1)*;*

Statement (2): $\boldsymbol{a}_i$ is not a local minimum of $L_i(\boldsymbol{x}_i)$ (i.e., $\boldsymbol{a}_i$ will not be the solution to (6.9)*) when ρ_i^t is finite.*

Proof: Statement (1) follows from Lemma 2.5 in [11] directly. The proof of Statement (2) can also be obtained following the proof of Lemma 2.5 in [11]. First let us denote $\sum\limits_{j \in \mathcal{N}_i} (\boldsymbol{\lambda}_{i,j}^{itT} \boldsymbol{x}_i + \frac{\rho_i^t}{2} \parallel \boldsymbol{x}_i - \boldsymbol{z}_{i,j}^t \parallel^2)$ by $g_i(\boldsymbol{x}_i)$. Then (6.14) can be written as

$$L_i(\boldsymbol{x}_i) = f_i(\boldsymbol{x}_i) + g_i(\boldsymbol{x}_i), \tag{6.15}$$

Note that $L_i(\boldsymbol{x}_i)$ is not differentiable at $\boldsymbol{a}_i$. Nonetheless, the directional derivative of L_i at every point $\boldsymbol{x}_i$ in the direction $\boldsymbol{v} \in \mathbb{R}^D$ exists and is given as follows [11]:

$$L_i'(\bar{\boldsymbol{x}}_i; \boldsymbol{v}) = \begin{cases} \triangledown L_i(\bar{\boldsymbol{x}}_i)^T \boldsymbol{v}, & \bar{\boldsymbol{x}}_i \neq \boldsymbol{a}_i, \\ \triangledown g_i(\boldsymbol{a}_i)^T \boldsymbol{v} - 2r_i \parallel \boldsymbol{v} \parallel, & \bar{\boldsymbol{x}}_i = \boldsymbol{a}_i. \end{cases} \tag{6.16}$$

Noting that g_i is differentiable at $\boldsymbol{a}_i$, if $\triangledown g_i(\boldsymbol{a}_i) \neq \mathbf{0}$, using (6.16) we have

$$L_i'(\bar{\boldsymbol{x}}_i; -\triangledown g_i(\boldsymbol{a}_i)) = - \parallel \triangledown g_i(\boldsymbol{a}_i) \parallel^2 - 2r_i \parallel \triangledown g_i(\boldsymbol{a}_i) \parallel < 0.$$

If $\triangledown g_i(\boldsymbol{a}_i) = \mathbf{0}$, then for every $\boldsymbol{v} \neq \mathbf{0}$ we have

$$L_i'(\bar{\boldsymbol{x}}_i; \boldsymbol{v}) = -2r_i \parallel \boldsymbol{v} \parallel < 0.$$

So there always exists a descent direction at $\boldsymbol{a}_i$. Therefore, $\boldsymbol{a}_i$ is not a local minimum of $L_i(\boldsymbol{x}_i)$. This concludes the proof of statement (2). ■

Next, we derive a closed-form solution (local minimum) to (6.9) by applying the zero-gradient condition [97] and the positive definite property of Hessian. First, let us find a solution to the zero-gradient condition:

$$\nabla f_i(\boldsymbol{x}_i^{t+1}) + \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^t + \rho_i^t(\boldsymbol{x}_i^{t+1} - \boldsymbol{z}_{i,j}^t)) = \mathbf{0}, \tag{6.17}$$

where

$$\nabla f_i(\boldsymbol{x}_i) = 2(\boldsymbol{x}_i - \boldsymbol{a}_i) - \frac{2r_i(\boldsymbol{x}_i - \boldsymbol{a}_i)}{\parallel \boldsymbol{x}_i - \boldsymbol{a}_i \parallel}, \quad \boldsymbol{x}_i \neq \boldsymbol{a}_i.$$

It is worth noting that here our purpose is to find one solution (not all) that satisfies both the zero-gradient condition [97] and the positive definite property of Hessian, i.e., one local minimum to (6.9).

Rewrite (6.17) as

$$\sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{x}_i^{t+1} - \boldsymbol{z}_{i,j}^t + \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t}) = 2(\boldsymbol{a}_i - \boldsymbol{x}_i^{t+1})(1 - \frac{r_i}{\parallel \boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i \parallel}). \tag{6.18}$$

Now, we provide a solution to (6.18) under two situations.

S1: when $\sum_{j\in\mathcal{N}_i}(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t}) - N_i\boldsymbol{a}_i \neq \mathbf{0}$ holds, let us set $\boldsymbol{x}_i^{t+1} = \boldsymbol{a}_i + \zeta\boldsymbol{\mu}$ with scalar $\zeta > 0$ and vector $\parallel \boldsymbol{\mu} \parallel = 1$, then it is clear that (6.18) can be rewritten as [97]

$$\sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) = \boldsymbol{\mu}[\zeta(2 + N_i\rho_i^t) - 2r_i]$$

if ζ and $\boldsymbol{\mu}$ are selected as follows

$$\zeta = \frac{2r_i + \parallel \sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) \parallel}{2 + N_i\rho_i^t},$$

$$\boldsymbol{\mu} = \frac{\sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i)}{\parallel \sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) \parallel}.$$

So we have that

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{a}_i + \frac{2r_i + \parallel \sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) \parallel}{2 + N_i\rho_i^t} \cdot \frac{\sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i)}{\parallel \sum_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) \parallel} \tag{6.19}$$

is a solution to (6.18), i.e., a stationary point of L_i.

S2: when $\sum\limits_{j\in\mathcal{N}_i}(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t}) - N_i\boldsymbol{a}_i = \boldsymbol{0}$ holds and ρ_i^t is a finite value, (6.18) is equal to

$$N_i\rho_i^t(\boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i) = 2(\boldsymbol{a}_i - \boldsymbol{x}_i^{t+1})(1 - \frac{r_i}{\parallel \boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i \parallel}). \tag{6.20}$$

It is easy to obtain that

$$\boldsymbol{x}_i^{t+1} \in \{\boldsymbol{a}_i + \frac{2r_i}{2 + N_i\rho_i^t}\boldsymbol{\varepsilon}\}, \quad \parallel \boldsymbol{\varepsilon} \parallel = 1 \tag{6.21}$$

is a solution to (6.18), i.e., a stationary point of L_i.

A commonly used way to choose $\boldsymbol{x}_i^{t+1}$ in this situation is to set $\boldsymbol{x}_i^{t+1} = \boldsymbol{a}_i + \frac{2r_i}{2+N_i\rho_i^t}\frac{\boldsymbol{x}_i^t - \boldsymbol{a}_i}{\|\boldsymbol{x}_i^t - \boldsymbol{a}_i\|}$.

Lemma 6. *The stationary points provided in* (6.19) *and* (6.21) *are also local minima of L_i, i.e., solutions to* (6.9) *(under situation S1 and situation S2, respectively).*

Proof: First, we prove that the point in (6.19) is a local minimum. Consider the Hessian of L_i (denoted by $\triangledown^2 L_i$), i.e.,

$$(2 + N_i\rho_i^t)I_D - \frac{2r_i}{\parallel \boldsymbol{x}_i - \boldsymbol{a}_i \parallel}I_D + \frac{2r_i(\boldsymbol{x}_i - \boldsymbol{a}_i)(\boldsymbol{x}_i - \boldsymbol{a}_i)^T}{\parallel \boldsymbol{x}_i - \boldsymbol{a}_i \parallel^3},$$

at $\boldsymbol{x}_i^{t+1}$ in (6.19), we have

$$\triangledown^2 L_i(\boldsymbol{x}_i^{t+1}) = \frac{2r_i(\boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i)(\boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i)^T}{\parallel \boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i \parallel^3} + (2 + N_i\rho_i^t)I_D - \frac{2r_i(2 + N_i\rho_i^t)}{2r_i + \parallel \sum\limits_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) \parallel}I_D. \tag{6.22}$$

Since $\sum\limits_{j\in\mathcal{N}_i}(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t}) - N_i\boldsymbol{a}_i \neq \boldsymbol{0}$, we have that

$$(2 + N_i\rho_i^t)I_D - \frac{2r_i(2 + N_i\rho_i^t)}{2r_i + \parallel \sum\limits_{j\in\mathcal{N}_i}\rho_i^t(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t} - \boldsymbol{a}_i) \parallel}I_D \tag{6.23}$$

is positive definite. In addition, because the matrix $\boldsymbol{y}\boldsymbol{y}^T$ is positive semidefinite, we have

$$\frac{2r_i(\boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i)(\boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i)^T}{\parallel \boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i \parallel^3} \tag{6.24}$$

is positive semidefinite. A positive definite matrix plus a positive semidefinite matrix is positive definite. So $\triangledown^2 L_i(\boldsymbol{x}_i^{t+1})$ is positive definite and therefore $\boldsymbol{x}_i^{t+1}$ in (6.19) is a local minimum of L_i.

For situation S2, first note that (6.14) can be rewritten as

$$\bar{L}_i(\boldsymbol{x}_i) = f_i(\boldsymbol{x}_i) + \frac{\rho_i^t N_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{b}_i^t \parallel^2 + c_i^t, \tag{6.25}$$

where $\boldsymbol{b}_i^t = \frac{\sum_{j \in \mathcal{N}_i} (\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t})}{N_i}$ and $c_i^t = -\frac{\rho_i^t N_i}{2} \parallel \boldsymbol{b}_i^t \parallel^2 + \sum_{j \in \mathcal{N}_i} \parallel \boldsymbol{z}_{i,j}^t \parallel^2$. When $\boldsymbol{b}_i^t = \boldsymbol{a}_i$ (situation S2), (6.25) can be rewritten as

$$\bar{L}_i(\boldsymbol{x}_i) = \frac{2 + \rho_i^t N_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{a}_i \parallel^2 - 2r_i \parallel \boldsymbol{x}_i - \boldsymbol{a}_i \parallel + r_i^2 + c_i^t. \tag{6.26}$$

Function (6.26) is a simple quadratic function and it is clear that the minimum is reached at points that satisfy $\parallel \boldsymbol{x}_i - \boldsymbol{a}_i \parallel = \frac{2r_i}{2+\rho_i^t N_i}$. Since $\boldsymbol{x}_i^{t+1}$ in (6.21) satisfies $\parallel \boldsymbol{x}_i^{t+1} - \boldsymbol{a}_i \parallel = \frac{2r_i}{2+\rho_i^t N_i}$, it is a local minimum of $\bar{L}_i$, i.e., a local minimum of L_i. ■

In Algorithm 5, after each sensor obtains its local estimated event position $\boldsymbol{x}_i^t$ in each iteration, it sends a copy of $\boldsymbol{x}_i^t$ to its neighboring sensors in $\mathcal{N}_i$. This guarantees the consistency of individual estimates across the entire network. In general, Algorithm 5 is highly scalable and flexible. However, if we consider the required storage space in each sensor, we can find that each sensor i has to store the following values: $\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{z}_{i,j}, \rho_i^t, \rho_j^t, \boldsymbol{\lambda}_{i,j}^i, \boldsymbol{\lambda}_{i,j}^j, N_i$, which require a storage space of $4DN_i + D + N_i + 2$ if we simply assume that any real number is stored by one storage cell. Note that these values are updated with iteration and hence can be demanding in terms of required storage space. In the following, we simplify the algorithm to save storage space in each sensor.

By substituting (6.12) and (6.13) into (6.11) [106], we get

$$\boldsymbol{z}_{i,j}^{t+1} = \frac{1}{\rho_i^t + \rho_j^t}(\rho_i^t \boldsymbol{x}_i^{t+1} + \rho_j^t \boldsymbol{x}_j^{t+1} + \boldsymbol{\lambda}_{i,j}^{i(t-1)} + \boldsymbol{\lambda}_{i,j}^{j(t-1)} + \rho_i^{t-1}(\boldsymbol{x}_i^t - \boldsymbol{z}_{i,j}^t) + \rho_j^{t-1}(\boldsymbol{x}_j^t - \boldsymbol{z}_{i,j}^t)). \tag{6.27}$$

Then by substituting (6.11) into $\boldsymbol{z}_{i,j}^t$ again, we get

$$\boldsymbol{z}_{i,j}^{t+1} = \frac{1}{\rho_i^t + \rho_j^t}(\rho_i^t \boldsymbol{x}_i^{t+1} + \rho_j^t \boldsymbol{x}_j^{t+1}), \tag{6.28}$$

$$\boldsymbol{\lambda}_{i,j}^{i(t+1)} + \boldsymbol{\lambda}_{i,j}^{j(t+1)} = 0. \tag{6.29}$$

From (6.28) and (6.29), we can simplify our algorithm by omitting the computation of $\boldsymbol{z}_{i,j}$ and $\boldsymbol{\lambda}_{i,j}^j, \forall j \in \mathcal{N}_i$ in sensor i. First, we introduce a vector $\boldsymbol{\lambda}_i = \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{i,j}^i$, which leads to (6.30) from the relationship in (6.12)

$$\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t + \sum_{j\in\mathcal{N}_i} \frac{\rho_i^t \rho_j^t}{\rho_i^t + \rho_j^t}(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}). \tag{6.30}$$

Then by substituting $\boldsymbol{z}_{i,j}^t$ in (6.19) with (6.28) and substituting $\sum\limits_{j\in\mathcal{N}_i} \boldsymbol{\lambda}_{i,j}^i$ in (6.19) with $\boldsymbol{\lambda}_i$, we have

$$\begin{aligned}\boldsymbol{x}_i^{t+1} = \boldsymbol{a}_i + &\frac{2r_i + \| \sum\limits_{j\in\mathcal{N}_i}(\frac{\rho_i^t}{\rho_i^{t-1}+\rho_j^{t-1}}(\rho_i^{t-1}\boldsymbol{x}_i^t + \rho_j^{t-1}\boldsymbol{x}_j^t) - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \rho_i^t \boldsymbol{a}_i) \|}{2 + N_i\rho_i^t} \\ &\cdot \frac{\sum\limits_{j\in\mathcal{N}_i}(\frac{\rho_i^t}{\rho_i^{t-1}+\rho_j^{t-1}}(\rho_i^{t-1}\boldsymbol{x}_i^t + \rho_j^{t-1}\boldsymbol{x}_j^t) - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \rho_i^t \boldsymbol{a}_i)}{\| \sum\limits_{j\in\mathcal{N}_i}(\frac{\rho_i^t}{\rho_i^{t-1}+\rho_j^{t-1}}(\rho_i^{t-1}\boldsymbol{x}_i^t + \rho_j^{t-1}\boldsymbol{x}_j^t) - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \rho_i^t \boldsymbol{a}_i) \|}\end{aligned} \tag{6.31}$$

under situation S1, i.e., when $\sum\limits_{j\in\mathcal{N}_i}(\frac{\rho_i^t}{\rho_i^{t-1}+\rho_j^{t-1}}(\rho_i^{t-1}\boldsymbol{x}_i^t + \rho_j^{t-1}\boldsymbol{x}_j^t) - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \rho_i^t \boldsymbol{a}_i) \neq \mathbf{0}$ holds.

The simplified algorithm is described in Algorithm 6.

Algorithm 6

Initial Setup: Each sensor initializes $\boldsymbol{x}_i^0$, $\boldsymbol{\lambda}_i^0$, ρ_i^0, and exchanges $\boldsymbol{x}_i^0$ with neighboring sensors.
Input: $\boldsymbol{x}_i^t$, $\boldsymbol{\lambda}_i^t$, ρ_i^t, ρ_i^{t-1}
Output: $\boldsymbol{x}_i^{t+1}$, $\boldsymbol{\lambda}_i^{t+1}$, ρ_i^{t+1}

1. Each sensor updates its local vector $\boldsymbol{x}_i^{t+1}$ in parallel according to the update rule in (6.31) or (6.21);
2. Each sensor sends its local vector $\boldsymbol{x}_i^{t+1}$ and ρ_i^t to neighboring sensors in $\mathcal{N}_i$;
3. Each sensor computes $\boldsymbol{\lambda}_i^{t+1}$ according to (6.30);
4. Each sensor updates $\rho_i^t \to \rho_i^{t+1}$. Detailed updating rules are given in Sec. 6.3;
5. Set $t = t+1$, and go to 1.

From Algorithm 6, we can see that each sensor has to store the following values: $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, ρ_i^t, ρ_j^t, ρ_i^{t-1}, ρ_j^{t-1}, $\boldsymbol{\lambda}_i$, N_i, which require a storage space of $DN_i + 2N_i + 2D + 3$ that is less than Algorithm 5. We remark that Algorithm 6 is a simplified version for Algorithm 5 in terms of the required storage space, and it differs from Algorithm 5 only by streamlining the steps in the conventional ADMM algorithm.

In addition, if we set $\rho_i^{t+1} = \rho_i^t = \bar{\rho}, \forall i \in \{1, 2, ..., N\}$, i.e., keep ρ_i as a constant, (6.30) and (6.31) can be simplified as:

$$\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t + \frac{\bar{\rho}}{2}(N_i\boldsymbol{x}_i^{t+1} - \sum_{j\in\mathcal{N}_i}\boldsymbol{x}_j^{t+1}),$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{a}_i + \frac{2r_i + \| \sum\limits_{j \in \mathcal{N}_i} (\frac{\bar{\rho}(\boldsymbol{x}_i^t + \boldsymbol{x}_j^t)}{2} - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \bar{\rho}\boldsymbol{a}_i) \|}{2 + N_i \bar{\rho}} \cdot \frac{\sum\limits_{j \in \mathcal{N}_i} (\frac{\bar{\rho}(\boldsymbol{x}_i^t + \boldsymbol{x}_j^t)}{2} - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \bar{\rho}\boldsymbol{a}_i)}{\| \sum\limits_{j \in \mathcal{N}_i} (\frac{\bar{\rho}(\boldsymbol{x}_i^t + \boldsymbol{x}_j^t)}{2} - \frac{\boldsymbol{\lambda}_i^t}{N_i} - \bar{\rho}\boldsymbol{a}_i) \|}$$

which require a even smaller storage space of $DN_i + 2D + 2$ in Algorithm 6. Furthermore, neighboring sensors only need to exchange local estimates of event position, which reduces the communication overhead. The convergence performance of the algorithm under a constant penalty parameter $\bar{\rho}$ is evaluated numerically in Sec. 6.4. Theoretical convergence analysis is detailed in the following section.

6.3 Convergence Analysis

Since f_i in the objective function is non-convex and non-continuously differentiable, no proof is currently available for the convergence of Algorithm 5 if ρ_i^t is kept constant (the same case for Algorithm 6), although simulation results show that the algorithm converges well under a constant ρ_i^t in Sec. 6.4. Inspired by the results in [7, 8, 28], we propose to update ρ_i^t in a time-varying way. Here, by time-varying we mean that the penalty parameter is updated at every iteration. We update the penalty parameter ρ_i^t according to the following rule:

If the following relationship holds in Algorithm 5

$$\| \sum_{j \in \mathcal{N}_i} (\boldsymbol{x}_i^{t+1} - \boldsymbol{z}_{i,j}^{t+1}) \|_\infty \leq \epsilon \| \sum_{j \in \mathcal{N}_i} (\boldsymbol{x}_i^t - \boldsymbol{z}_{i,j}^t) \|_\infty, \tag{6.32}$$

or accordingly the following relationship holds in Algorithm 6

$$\| \sum_{j \in \mathcal{N}_i} \frac{\rho_j^t}{\rho_i^t + \rho_j^t} (\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}) \|_\infty \leq \epsilon \| \sum_{j \in \mathcal{N}_i} \frac{\rho_j^{t-1}}{\rho_i^{t-1} + \rho_j^{t-1}} (\boldsymbol{x}_i^t - \boldsymbol{x}_j^t) \|_\infty, \tag{6.33}$$

then we update ρ_i^{t+1} as

$$\rho_i^{t+1} = \max\{\rho_j^t, j \in \mathcal{N}_i \cup \{i\}\}, \tag{6.34}$$

otherwise, we update ρ_i^{t+1} as

$$\rho_i^{t+1} = \eta(\max\{\rho_j^t, j \in \mathcal{N}_i \cup \{i\}\}), \tag{6.35}$$

where ϵ resides in the interval $[0,1)$, η is larger than but close to 1. Here, $\| \sum_{j\in\mathcal{N}_i} (\boldsymbol{x}_i^{t+1} - \boldsymbol{z}_{i,j}^{t+1}) \|_\infty$ is a measure of the local primary gap. In the condition based update rule, (6.34) is used to keep the penalty parameter ρ_i as consistent throughout the network as possible when the local primary gap decreases sufficiently, i.e., when (6.32) holds, whereas (6.35) is used to reinforce the constraint $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ in the next update of $\boldsymbol{x}_i$ if the local primary gap does not decrease sufficiently enough. According to (6.9), we can see that a larger ρ implies a stronger influence of the constraint $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ when updating $\boldsymbol{x}_i$.

In addition, the Lagrange multipliers and the search domain for $\boldsymbol{x}_i^{t+1}$ are required to be bounded [7, 8, 28]. Therefore, we substitute the update rule for Lagrange multipliers in (6.30) with the following form:

$$\boldsymbol{\lambda}_i^{t+1} = \mathcal{P}_\Omega[\boldsymbol{\lambda}_i^t + \sum_{j\in\mathcal{N}_i} \frac{\rho_i^t \rho_j^t}{\rho_i^t + \rho_j^t}(\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1})], \tag{6.36}$$

where $\mathcal{P}_\Omega$ denotes the projection on the interval $\Omega = [\boldsymbol{\lambda}_{\text{min}}, \boldsymbol{\lambda}_{\text{max}}]$, i.e.,

$$\mathcal{P}_\Omega[\boldsymbol{\lambda}]_j = \begin{cases} [\boldsymbol{\lambda}_{\text{min}}]_j, & \text{if } [\boldsymbol{\lambda}]_j \leq [\boldsymbol{\lambda}_{\text{min}}]_j \ ; \\ [\boldsymbol{\lambda}_{\text{max}}]_j, & \text{if } [\boldsymbol{\lambda}]_j \geq [\boldsymbol{\lambda}_{\text{max}}]_j \ ; \\ [\boldsymbol{\lambda}]_j, & \text{otherwise,} \end{cases}$$

and $[\boldsymbol{\lambda}]_j$ is the jth element of vector $\boldsymbol{\lambda}$. To prevent unwanted clippings, $\boldsymbol{\lambda}_{\text{max}}$ should be large enough ($\boldsymbol{\lambda}_{\text{min}}$ should be small enough).

For the update rule in (6.17), we add a constraint set

$$\mathcal{X}_i = \{\boldsymbol{x}_i | f_i(\boldsymbol{x}_i) \leq f(\bar{\boldsymbol{x}})\},$$

where $\bar{\boldsymbol{x}}$ is some initial estimate such that $f(\bar{\boldsymbol{x}}) \leq \infty$ is true. Therefore, (6.17) can be substituted by

$$\mathcal{P}_{\mathcal{X}_i}[\boldsymbol{x}_i^{k+1} - G_i(\boldsymbol{x}_i^{k+1})] - \boldsymbol{x}_i^{k+1} = \boldsymbol{0}, \tag{6.37}$$

where $G_i(\boldsymbol{x}_i^{k+1})$ denotes the left side function of (6.17). Then we need to modify the update rule of $\boldsymbol{x}_i^{t+1}$ for situation S1. Denote the solutions in (6.19) and (6.31) as $\dot{\boldsymbol{x}}_i^{t+1}$, we have

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{a}_i + \frac{(r_i+\sqrt{f(\overline{\boldsymbol{x}})})(\dot{\boldsymbol{x}}_i^{t+1}-\boldsymbol{a}_i)}{\|\dot{\boldsymbol{x}}_i^{t+1}-\boldsymbol{a}_i\|}, & \dot{\boldsymbol{x}}_i^{t+1} \notin \mathcal{X}_i; \\ \dot{\boldsymbol{x}}_i^{t+1}, & \dot{\boldsymbol{x}}_i^{t+1} \in \mathcal{X}_i. \end{cases} \tag{6.38}$$

We first introduce two conclusions in [7] and [8], which help to conclude our theorem. (Note that Algorithm 5 and Algorithm 6 using the rules in (6.32)-(6.38) are designed following the idea of Algorithm 3.1 in [7] and Algorithm 2 in [8].)

Lemma 7. *(Theorem 3.2 in [8]) Let $\{\boldsymbol{x}^t\}$ be a sequence generated by Algorithm 5 or Algorithm 6. Then at least one of the following possibilities hold when Algorithm 5 and Algorithm 6 are updated following the rules in* (6.32)-(6.38)*: 1. The sequence admits a feasible limit point; 2. The sequence admits an infeasible degenerate limit point.*

Lemma 8. *(Theorem 4.1 in [7]) Let $\{\boldsymbol{x}^t\}$ be a sequence generated by Algorithm 5 or Algorithm 6 using the rules in* (6.32)-(6.38)*. Let $\boldsymbol{x}^*$ be a limit point of $\{\boldsymbol{x}^t\}$. If the sequence of penalty parameter $\rho_i^t, \forall i \in \{1, 2, ..., N\}$ in* (6.35) *is bounded, the limit point $\{\boldsymbol{x}^t\}$ is feasible.*

Now we can conclude the convergence properties of our algorithms.

Theorem 13. *If the Lagrange multipliers and penalty parameters in Algorithm 5 and Algorithm 6 are updated following the rules in* (6.32)-(6.38)*, Algorithm 5 and Algorithm 6 are guaranteed to admit a limit point $\boldsymbol{x}^*$ for any parameter choice, provided that they are chosen in the required range, i.e., $\rho_i^0 > 0$, $\eta > 1$, $0 \leq \epsilon < 1$, $\boldsymbol{\lambda}_{\max} > \boldsymbol{0}$, and $\boldsymbol{\lambda}_{\min} < \boldsymbol{0}$. In addition, the limit point is admissible provided that $\lim_{t\to\infty} \rho_i^t = \hat{\rho} < \infty, \forall i \in \{1, 2, ..., N\}$ is true, i.e., all ρ_i^t are bounded, and the Lagrange multipliers are not clipped.*

Proof: The proof of Theorem 13 follows from conclusions in [7, 8]. First, note that Algorithm 5 and Algorithm 6 using the rules in (6.32)-(6.38) are designed following the idea of Algorithm 3.1 in [7] and Algorithm 2 in [8], where $\boldsymbol{x}$ belongs to the set $\{\mathcal{X}_1 \times, ..., \times \mathcal{X}_N\}$. Then according to Theorem 3.2 in [8], we can get that Algorithm 5 and Algorithm 6 admit a limit point $\boldsymbol{x}_*$. It is worth noting that in [8], the derivative of the objective function (denoted by $F(\boldsymbol{x})$ in [8]) should be continuous, while in our case, $f_i(\boldsymbol{x}_i)$ is non-differentiable at $\boldsymbol{x}_i = \boldsymbol{a}_i$. However, this will not be a problem in our case, as explained as follows. From the proof of Theorem 3.2 in [8], we can see that the continuity condition is used to guarantee $\| \lim_{t\to\infty} \frac{F(\boldsymbol{x}^t)}{\|\rho^t\|_\infty} \| = 0$, when $\| \boldsymbol{\rho}^t \|_\infty$ tends to ∞ as $t \to \infty$. Recalling Lemma 5, it is clear that the only possible situation that we will have $\boldsymbol{x}_i^{t+1}$ tending to $\boldsymbol{a}_i$ is when $\sum\limits_{j\in\mathcal{N}_i}(\boldsymbol{z}_{i,j}^t - \frac{\boldsymbol{\lambda}_{i,j}^t}{\rho_i^t}) - N_i\boldsymbol{a}_i = \mathbf{0}$ holds and ρ_i^t tends to ∞. And in this situation, we have $\| \lim\limits_{\boldsymbol{x}_i^t\to\boldsymbol{a}_i} \nabla f_i(\boldsymbol{x}_i^t) \| = 2r_i$, which still guarantees $\| \lim\limits_{t\to\infty} \frac{F(\boldsymbol{x}^t)}{\|\rho^t\|_\infty} \| = 0$. So Theorem 3.2 in [8] holds for Algorithm 5 and Algorithm 6. When all ρ_i^t are bounded, and the Lagrange multipliers are not clipped, according to Theorem 4.1 in [7], $\boldsymbol{x}_*$ is a feasible limit point. This concludes the proof of Theorem 13. ■

In general, ρ_i^0 should be set to a small value no more than 1 and η should be set slightly larger than but close to 1, since a large ρ_i^t will slow down the convergence rate. The parameter ϵ resides in the interval $[0, 1)$, but it is suggested to be chosen slightly smaller than but close to 1 [97]. To prevent unwanted clippings, $\boldsymbol{\lambda}_{\text{max}}$ should be large enough and $\boldsymbol{\lambda}_{\text{min}}$ should be small enough.

Remark 17. *An important contribution of our algorithm with respect to the centralized optimization approaches in [7] and [8] is that our algorithm can be implemented in a completely decentralized manner. In addition, [7] and [8] address smooth (continuously differentiable) objective functions whereas this chapter extends the results to address the non-smooth event localization problem where the objective function is not always differentiable.*

Remark 18. *It is worth noting that the target event localization problem considered here is different from the self-cooperative localization problem in [28, 59, 106] where sensors with unknown positions are embedded with computation capability to estimate their own positions by themselves. The differences are evident from the following example. Suppose that there is only one target to localize. In the case of [28, 59, 106], the target will be a sensor with unknown position and it estimates its own position* alone *in a* centralized *way based on all information gathered from adjacent sensors, including their positions and corresponding range measurements. Whereas in our case, the target is an event without any communication or computation capability and the event position estimation process is conducted* cooperatively *in a* decentralized *way among the sensors.*

6.4 Numerical Simulations

In this section, we first illustrated the effectiveness of the proposed algorithm under both time-varying and constant penalty parameters. Then we compared our Algorithm 6 with existing results including the PONLM algorithm [101], the PPM algorithm [56], and the DAPA algorithm [135]. It is worth noting that as indicated in Sec. 6.2, Algorithm 6 differs from Algorithm 5 only by streamlining the steps in the conventional ADMM algorithm. Therefore, Algorithm 5 and Algorithm 6 have the same performance. The Matlab code for all simulations in this section can be found in [1].

The following two performance indices $\mathrm{ERR}_{\mathrm{RMSE}}$ and $\mathrm{INC}_{\mathrm{RMSE}}$ are defined below for the convenience of performance comparison:

Localization Error: we use the root mean square error (RMSE) to quantify the error between estimated and true positions of the target event for every sensor, which is denoted as $\mathrm{ERR}_{\mathrm{RMSE}}$:

$$\mathrm{ERR}_{\mathrm{RMSE}} = \sqrt{\frac{\sum_{j=1}^{L} \| \boldsymbol{x}_j - \boldsymbol{x}^* \|^2}{L}},$$

where L is the number of Monte Carlo trials, $\boldsymbol{x}_j$ is the estimated position in the jth Monte Carlo trial in a certain sensor, and $\boldsymbol{x}^*$ is the true position of the target event.

Localization Inconsistency: We also use the root mean square error (RMSE) to quantify the localization inconsistency (difference) in estimated event positions between N sensors, which is denoted as $\mathrm{INC}_{\mathrm{RMSE}}$:

$$\mathrm{INC}_{\mathrm{RMSE}} = \sqrt{\frac{\sum_{k=1}^{L} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \| \boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k} \|^2}{L}},$$

where L is the number of Monte Carlo trials, $\boldsymbol{x}_{i,k}$ is the estimated position obtained from the ith sensor in the kth Monte Carlo trial. Parameter N is the number of sensors.

6.4.1 Evaluation of the Proposed Algorithm with Respect to Penalty Parameters

A. The Convergence of Time-varying $\boldsymbol{\rho}$*:* As indicated in [121], the number of sensors required to achieve unique source identification is between 4 and 6, so we simulated Algorithm 6 under an event localization setup with five sensors and a target event located at $[7.5; 7.5]$. We set the initial values of time-varying penalty parameters as $\rho_i^0 = 1, \forall i \in \{1, 2, ..., N\}$, and other parameters as $\epsilon = 0.99$, $\eta = 1.0001$, respectively. In addition, when $\| \sum_{j \in \mathcal{N}_i} \frac{\rho_j^t}{\rho_i^t + \rho_j^t} (\boldsymbol{x}_i^{t+1} - \boldsymbol{x}_j^{t+1}) \|_\infty \leq 10^{-12}$ holds, $\boldsymbol{\rho}$ stops iterating. We ran the simulation for 1000 times with the sensors positions randomly chosen from $[-15, 15] \times [-15, 15]$. Simulation results showed that in all runs ρ_i converged to some finite value, no matter whether the target event was in or outside the convex hull of the five sensors. Fig. 6.2 visualizes the evolution of ρ_i in one specific run where the sensor positions are illustrated in Fig. 6.1 (we used the noise-free range measurements in this part).

B. Time-varying vs Constant Penalty Parameters: Because a constant penalty parameter can reduce

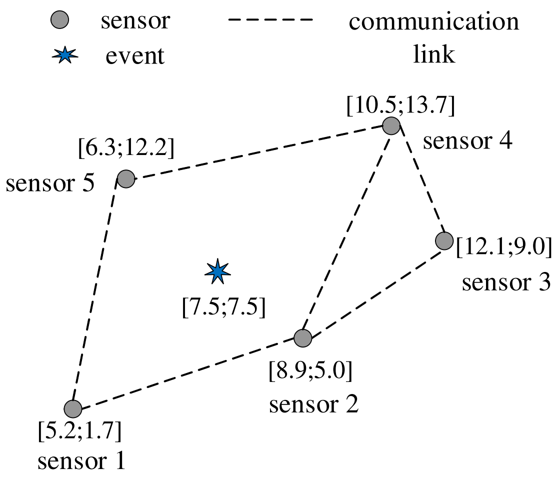


Figure 6.1: Event localization setup used in one of the simulation runs. The values in [•] denote positions (x, y coordinates) of sensors.

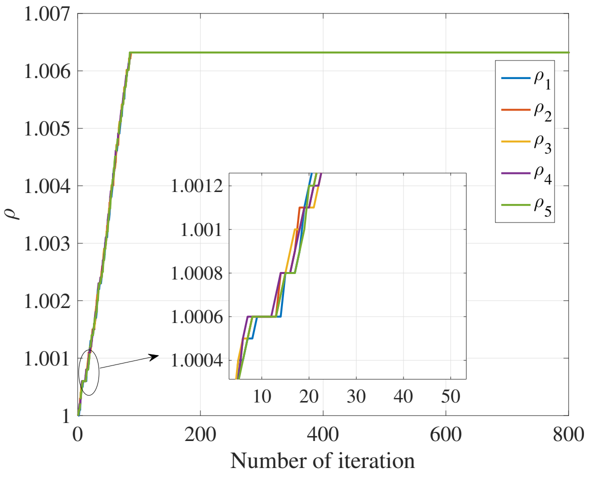
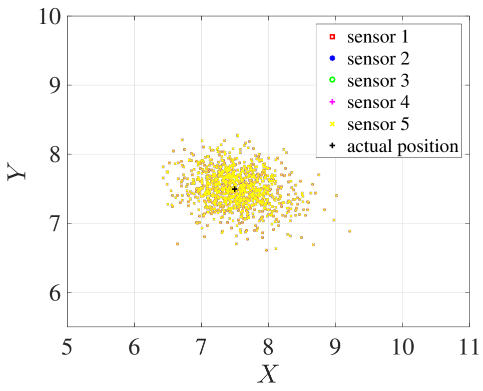
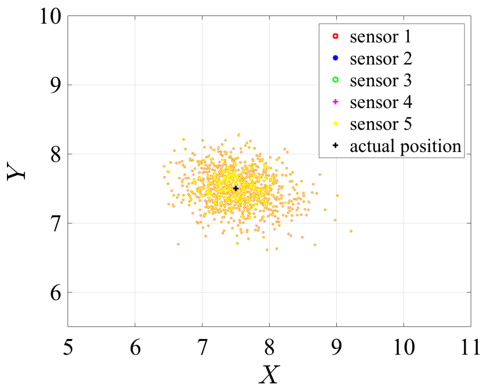


Figure 6.2: The evolution of ρ_i.

communication overhead and the storage space in each sensor, we simulated Algorithm 6 under a constant penalty parameter $\bar{\rho} = 1$. Fig. 6.3 visualizes the distribution of estimated event positions from 1000 Monte Carlo trials under a constant penalty parameter $\bar{\rho} = 1$ (Fig. 6.3b) and a time-varying $\boldsymbol{\rho}$ (Fig. 6.3a) with sensor positions given in Fig. 6.1. The Gaussian standard noise deviation (measurement noise) was set to $\sigma_i = 0.5$. The initial estimate $\boldsymbol{x}_i^0$ was randomly chosen in $[-100, 100] \times [-100, 100]$ in each Monte Carlo trial. It can be seen that performances are similar in the two cases.



(a) Time-varying $\boldsymbol{\rho}$



(b) Constant $\bar{\rho}$

Figure 6.3: Estimated event positions distribution for 1000 Monte Carlo trials under the measurement noise $\sigma_i = 0.5$.

We also evaluated the influence of the noise level on localization inconsistency of Algorithm 6 under time-varying and constant penalty parameters. The results are summarized in Table 6.1 where each data point is an average of 1000 Monte Carlo trials. They confirm that the proposed algorithm can achieve a good consistency across the entire sensor network even under large measurement noises.

Table 6.1: INC_{RMSE} of Algorithm 6 under different levels of measurement noise

σ_i	Time-varying $\boldsymbol{\rho}$	Constant $\bar{\rho}$
0.01	6.80×10^{-15}	2.42×10^{-15}
0.2	6.62×10^{-15}	2.46×10^{-15}
0.5	6.64×10^{-15}	2.50×10^{-15}
1	6.66×10^{-15}	2.61×10^{-15}
1.5	0.3858	0.3042
2	0.9778	0.8785

6.4.2 Comparison with Projection-based Algorithms

In this subsection, we compared the proposed Algorithm 6 with the PONLM algorithm [101], the DAPA algorithm [135], and the PPM algorithm [56]. For our algorithm, we set the initial values of parameters as $\rho_i^0 = 1, \forall i \in \{1, 2, ..., N\}$, $\epsilon = 0.99$, and $\eta = 1.0001$ respectively. For DAPA, we set the same parameters as in [135]: $\alpha_i = \frac{1}{t+2}, \beta_i = \frac{1}{t+1}, b_i = 1, \xi_i = 3, \forall i \in \{1, 2, ..., N\}$. For PONLM, since it is sequential, we set its updating order as $1 \to 2 \to ... \to N$. In addition, we made an equivalent transform from energy measurements to range measurements for DAPA and PONLM in our simulations. Note that PPM is a centralized algorithm and it requires a central node to collect and average local estimates from all sensors. Therefore, in fact, PPM is not applicable to our event localization, but we still list its results here so that the performance of our algorithm can be evaluated in context.

A. Convergence Performance: We used the event localization setup in Fig. 6.1 to compare the convergence performance. Fig. 6.4 visualizes the evolution of the localization error with iteration time t. The measurement noise was set to $\sigma_i = 0.01$, and each data point in Fig. 6.4 was an average of 1000 Monte Carlo trials. The settings for initial estimates, iteration times, and random noisy range measurements were identical for all algorithms in each Monte Carlo trial. From Fig. 6.4, we can see that Algorithm 6 has a comparable performance with the centralized method PPM in terms of localization accuracy and convergence rate. *It is worth noting that although PONLM has the highest convergence rate, it requires sensors to update their local estimates sequentially according to a globally predefined order. Therefore, it requires a much longer absolute updating time than all the other three algorithms, especially when the number of sensors is large.*

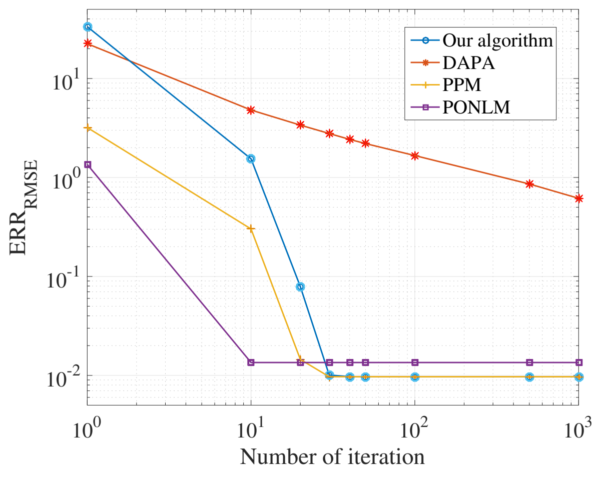


Figure 6.4: The evolution of localization error under the event localization setup in Fig. 6.1.

We also compared the convergence performance of our algorithm, PONLM, DAPA, and PPM in a network of 50 sensors randomly placed in the field $[-25, 25] \times [-25, 25]$. The sensors are assumed to communicate only with two neighbors to form a ring network. Fig. 6.5 visualizes the evolution of the localization error with iteration time t. The measurement noise was set to $\sigma_i = 0.5$, and each data point in Fig. 6.5 was an average of 500 Monte Carlo trials. We can see that our Algorithm 6 still has a comparable accuracy with the centralized version PPM while having a much faster convergence speed than DAPA. It is worth noting that although PONLM has a high convergence rate, it requires sensors to update their local estimates sequentially according to a globally predefined order. Therefore, it requires a much longer absolute updating time than all the other three algorithms.

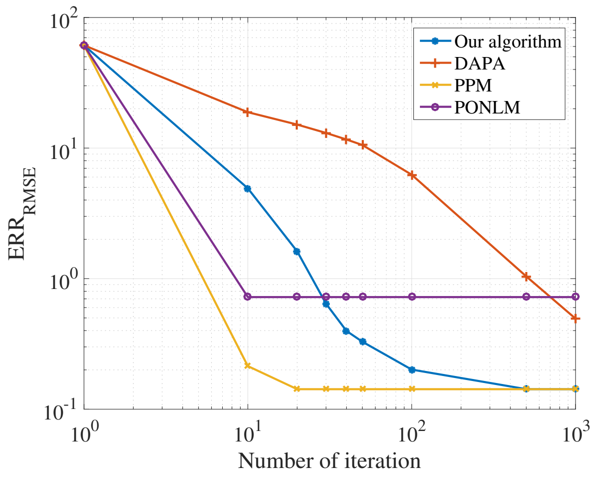


Figure 6.5: The evolution of localization error in a ring network of 50 sensors

.

B. The Influence of Measurement Noise on Localization Error: We also varied the level of measurement noise to check their influence on the localization error under the event localization setup in Fig. 6.1. The iteration time was set to 1000. Fig. 6.6 summarizes the ERR_{RMSE} of these algorithms under different σ_i with each data point being an average of 1000 Monte Carlo trials. It can be seen that Algorithm 6 has a comparable performance on the localization accuracy with the centralized method PPM. Furthermore, compared with PONLM and DAPA, our proposed algorithm showed consistently better performance under different levels of measurement noise. Moreover, when we used noise-free range measurements, the localization error of Algorithm 6 is on the level of 10^{-15}, which indicates that the algorithm can converge to the true target event position.

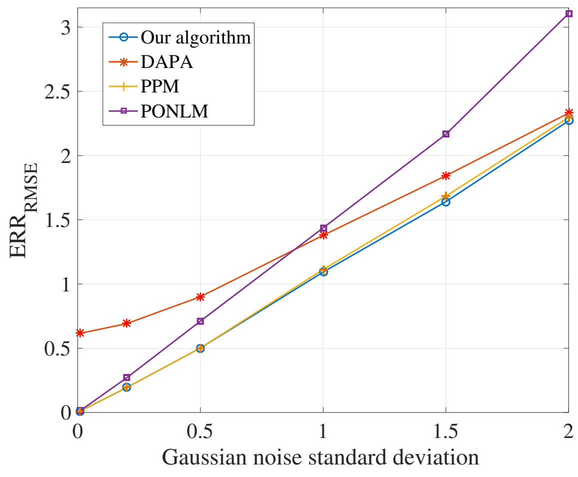


Figure 6.6: The influence of measurement noise on localization error.

C. The Influence of Initialization Settings: We randomly set different initial estimates to check its influence to the four algorithms. We considered two situations: when the target event is in the convex hull of sensors as illustrated in Fig. 6.1 and when the target event is outside the convex hull of sensors as illustrated in Fig. 6.7. The event localization setup in Fig. 6.7 is inspired by the practical acoustic event localization system in [18, 90].

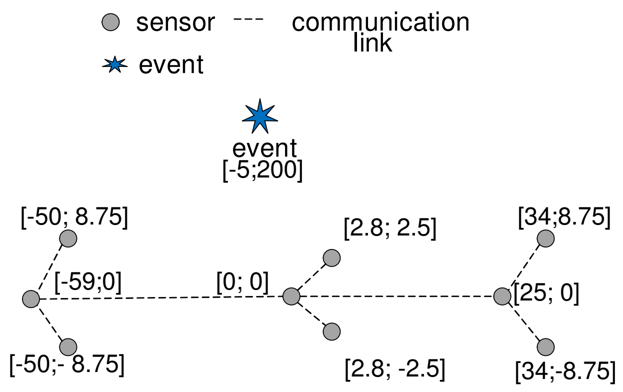


Figure 6.7: Event localization setup used in simulations. The values in [•] denote positions (x, y coordinates) of sensors.

When the target event is in the convex hull of sensors, all algorithms converged to the true target event position irrespective of the initial estimate $\boldsymbol{x}_i^0$. However, when the target event is outside the convex hull of sensors, DAPA did not converge to the true target event position in our simulations even if we set the initial

Table 6.2: Algorithm 6 vs PPM under random initial estimates

Initial Estimates	Algorithm 6			PPM		
	V_1	V_2	Correct Hits	V_1	V_2	Correct Hits
$\mathcal{Y}_3$	9826	174	98.26%	8423	1577	84.23%
$\mathcal{Y}_2$	9941	59	99.41%	8766	1234	87.66%
$\mathcal{Y}_1$	9991	9	99.91%	9397	603	93.97%

estimates close to the true target event position. For example, it converged to $[-15; 200]$ after 10^5 iterations when $\boldsymbol{x}_i^0 = [-20; 180], \forall i \in \{1, 2, ..., 9\}$, while Algorithm 6 and PPM converged to $[-5; 200]$ after 200 iterations. The failure of DAPA is understandable given that it always pursues the intersections of sensing rings, which do not necessarily coincide with the true target event location. Table 6.2 gives the number of times that Algorithm 6 and PPM converged to the true target event position $[-5; 200]$ (denoted as V_1) and another local minimum $[-4.9; -200]$ (denoted as V_2) when the initial estimates of the nine sensors were randomly chosen in the following three ranges in 10000 Monte Carlo trials, respectively: $\mathcal{Y}_1 := \{\boldsymbol{x} \in \mathbb{R}^2 | \parallel \boldsymbol{x} - [-5; 200] \parallel \leq 300\}$, $\mathcal{Y}_2 := \{\boldsymbol{x} \in \mathbb{R}^2 | \parallel \boldsymbol{x} - [-5; 200] \parallel \leq 400\}$, and $\mathcal{Y}_3 := \{\boldsymbol{x} \in \mathbb{R}^2 | \parallel \boldsymbol{x} - [-5; 200] \parallel \leq 500\}$. It can be seen that our proposed algorithm is more likely to converge to the true target event position. PONLM is not applicable to the event localization setup in Fig. 6.7, since there exists no path that connects all nine sensors in succession.

D. The Influence of Topology Changes: We simulated the influence of topology changes to Algorithm 6, the PONLM algorithm [101], and the DAPA algorithm [135] under the event localization setup in Fig. 6.1. We supposed that each communication link in Fig. 6.1 has a fixed probability of packet loss in each iteration, denoted as P. For Algorithm 6 and DAPA, if the packet on a communication link between two senors is lost, then these two sensors can not exchange local estimates in this iteration. Specifically for Algorithm 6, if a sensor does not receive any local estimates from its neighbors in some iteration time, it updates its local estimate as $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t$. For PONLM, if there exists one packet lose in the multi-hop path $1 \rightarrow 2 \rightarrow ... \rightarrow 5$, the update process is failed in this iteration. Therefore, the topology might change from iteration to iteration. Fig. 6.8 visualizes the evolution of localization error of the three algorithms under the packet loss probability of $P = 5\%$. The measurement noise was set to $\sigma_i = 0.01$, and each data point was an average of 1000 Monte Carlo trials. We can see that Algorithm 6 has the smallest localization error in the presence of topology changes caused by link failures. Fig. 6.9 visualizes the evolution of localization error of Algorithm 6 under packet loss probabilities of $P = 0\%$, $P = 5\%$, $P = 10\%$, $P = 15\%$, $P = 30\%$, and $P = 50\%$, respectively. It can be seen that Algorithm 6 reached high localization accuracy even under a high probability of packet loss, which indicates that Algorithm 6 is robust to network topology changes.

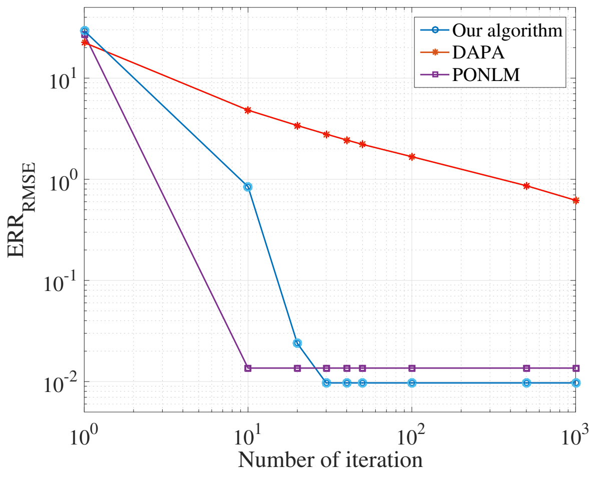


Figure 6.8: The evolution of localization error of our algorithm, PONLM, and DAPA under a packet loss probability of $P = 5\%$.

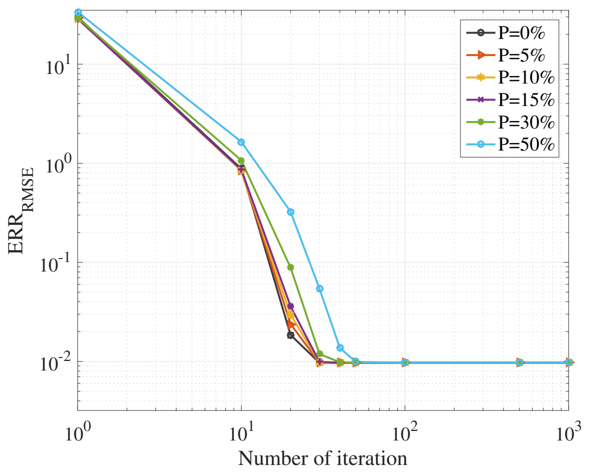


Figure 6.9: The evolution of localization error of Algorithm 6 under different packet loss probabilities.

Remark 19. *In Algorithm 5 and Algorithm 6, the update of $\boldsymbol{x}_i^t$ (local target event position estimate) at sensor i is influenced by the local estimates from its neighbors due to the constraint $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$. The constraints $\boldsymbol{x}_i = \boldsymbol{z}_{i,j}$ and $\boldsymbol{x}_j = \boldsymbol{z}_{i,j}$ drive the consistency of local estimates among neighboring sensors once they can communicate. So the consistency among local estimates of all sensors can still be guaranteed whenever neighboring sensors can exchange information under a connected communication pattern. This is in some*

Table 6.3: Complexity comparison among Algorithm 6, PONLM, and DAPA

	Algorithm 6		PONLM	DAPA
	Time-varying $\boldsymbol{\rho}$	Constant $\bar{\rho}$		
Communication Load	$\sum_{i=1}^{N}(D+1)N_i$	$\sum_{i=1}^{N} DN_i$	ND	$\sum_{i=1}^{N} DN_i$
Local Storage Space	$DN_i + 2N_i + 2D + 3$	$DN_i + 2D + 2$	$2D$	$DN_i + 3D + N_i + 2$
Execution Time (every 1000 iterations)	0.02s	0.0066s	0.002s	0.008s

sense similar to the asynchronous ADMM, whose convergence is guaranteed for convex function [119]. Our simulation results suggest a potential application of asynchronous ADMM in some non-convex and non-smooth functions.

E. Complexity Comparison: Table 6.3 provides a comparison on the overall network communication load per iteration, the required local storage space in each sensor, and the average computation complexity per sensor per iteration of Algorithm 6, the PONLM algorithm [101], and the DAPA algorithm [135]. We simply assume that any real number is stored by one storage cell. For computation complexity comparison, we used the average execution time that each sensor spends for 1000 iterations. From Table 6.3, we can see that Algorithm 6 with constant $\bar{\rho}$ has comparable performance with DAPA. In addition, although algorithm PONLM has the smallest communication overhead and local storage requirement, it requires sensors to update their local estimates sequentially according to a globally predefined order and is not amendable to parallelism.

F. The influence of the number of sensors: In this part, we simulated the influence of the number of sensors on our algorithm's convergence rate and localization error. We considered 50 sensors randomly placed in the field $[-25, 25] \times [-25, 25]$. We formed a ring network by randomly choosing 5, 10, 25, and 50 sensors, respectively. Fig. 6.10 visualizes the evolution of localization error of Algorithm 6 under these ring networks. The measurement noise was set to $\sigma_i = 0.5$, and each data point in Fig. 6.10 was an average of 500 Monte Carlo trials. Fig. 6.10 shows that with an increase in the number of sensors, the localization error decreases monotonically while the convergent speed is only moderately affected in a non-monotonic way. Therefore, if accuracy is of concern, then more sensors should be deployed. In addition, simulation results showed that the probability of acquiring the true target event position was always 100% for the three cases of 10 sensors, 25 sensors, and 50 sensors, respectively (cf. Table 6.4), which is higher than the case with 5 sensors. This is intuitive as the target event is more likely to be within the convex hull of all sensors when the number of sensors increases.

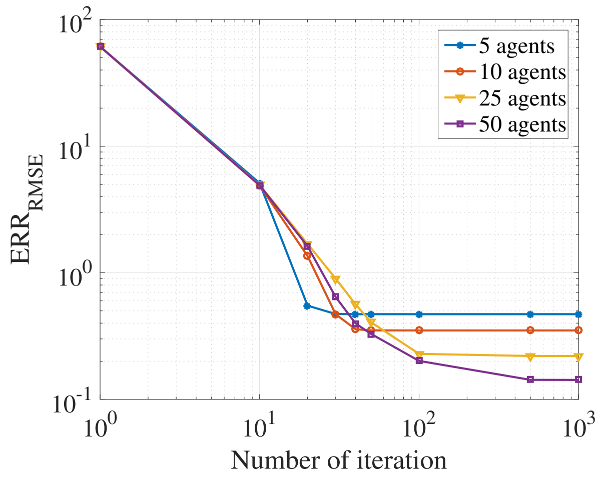


Figure 6.10: The evolution of localization error of Algorithm 6 under different numbers of sensors.

Table 6.4: Correct hits of Algorithm 6 under a ring topology composed of different numbers of sensors.

Number of Sensors	Correct Hits	Total Trials	Percentage
5	973	1000	97.3%
10	1000	1000	100%
25	1000	1000	100%
50	1000	1000	100%

6.5 Summaries

In this chapter, we proposed a completely decentralized parallel algorithm which solves the non-convex and non-smooth event localization problem directly without using convex relaxation. Simulation results confirm that our algorithm has better localization accuracy compared with other projection-based algorithms when the target event is in the convex hull of sensors. When the target event is outside the convex hull of sensors, our algorithm has a higher probability to converge to the right target event position than existing results. In addition, numerical simulations show that our algorithm has higher localization accuracy than existing approaches even in the presence of topology changes.

Chapter 7

Distributed Event Localization via ADMM

This chapter is motivated by acoustic event localization which is crucial on battlefields [25]. In such applications, the target event has no communication or computation capability, which differentiates the problem from sensor localization problems in which the locations of sensors are estimated [106]. Furthermore, in such applications, the target events lie outside the convex hull of deployed sensors, which renders existing projection-based algorithms inappropriate. SDP relaxation based algorithms can avoid the convex hull problem and are traditionally employed to solve the event localization problem [30, 91, 92, 113, 114, 124, 126]. However, as far as we known, existing SDP relaxation based algorithms for event localization are all centralized, with a central node collecting and processing all data, which makes them susceptible to processing center failure and traffic bottleneck. In this chapter, we propose two distributed event localization approaches based on a clustered architecture motivated by mobile acoustic localization applications such as the PinPoint$^{\text{TM}}$ system from BioMimetics Systems Inc. The PinPoint$^{\text{TM}}$ mobile localization sensor network can be deployed as a mobile infrastructure for impulsive threat event detection and localization [18, 25]. Each PinPoint$^{\text{TM}}$ sensor is a small omnidirectional microphone array which localizes impulsive acoustic events by correlating the ToA measurements among its microphone cells. In fact, since each sensor has an integrated microphone array, individual sensors are able to identify and localize a target event without assistance or cooperation with other sensors. However, due to close distances between the microphone cells, the accuracy of individual sensors is very limited and unsatisfactory, and collaboration among the sensors is necessary to improve localization accuracy [18, 25].

The above application motivated us to assume a localization architecture in which an entire network is divided into several clusters. A cluster head (which can be a regular sensor) collects and fuses measurements (e.g., noisy ranges) obtained from all members in its cluster. Two cluster heads in different clusters can exchange information (the local estimates of target events) if a communication link is available between them; otherwise they don't have access to each other's information. Based on the alternating direction method of multipliers (ADMM), we propose two scalable distributed algorithms named GS-ADMM and J-ADMM which do not require the target event to be within the convex hull of the deployed sensors. Our developed algorithms can also be applied in some other applications where a cluster-based architecture is employed. A typical example is the wide-area monitoring and control in large-scale power systems [83], [35]. To estimate the electro-mechanical oscillation modes, a large number of phasor measurement units (PMU) have to be deployed across a power network to conduct measurements. The measurements from PMUs have to be fused to diagnose the inter-area oscillation modes. However, wide-area communication between PMUs is very expensive [43]. To fuse information across the PMUs without imposing heavy communication overhead, a similar structure as ours is adopted in [83], [35]. Other examples on cluster-based architecture can be found in [55, 95, 109, 133].

The main contribution of this chapter is two ADMM-based distributed event localization algorithms, i.e., GS-ADMM and J-ADMM. Compared with existing centralized SDP relaxation based algorithms for event localization, the two algorithms divide the computation on a central node to different clusters to avoid possible center failure and traffic bottleneck, and in the mean time, guarantee consistency of the estimates across all clusters among which only limited communications are available. Furthermore, the two algorithms take advantages of SDP relaxation to avoid the convex hull problem compared with existing projection-based algorithms. Moreover, the algorithms are proven to converge with a convergence rate of $O(1/t)$ where t is the iteration time.

The rest of this chapter is organized as follows: Sec. 7.1 states the formulation of the problem. To solve the problem, a convex relaxation is required and the method proposed by [106] is recapitulated in Sec. 7.2. In Sec. 7.3 , two algorithms named GS-ADMM and J-ADMM are proposed based on ADMM, with their convergence properties analyzed in Sec. 7.4. Sec. 7.5 gives numerical simulation results. In the end, a summary is made in Section 7.6.

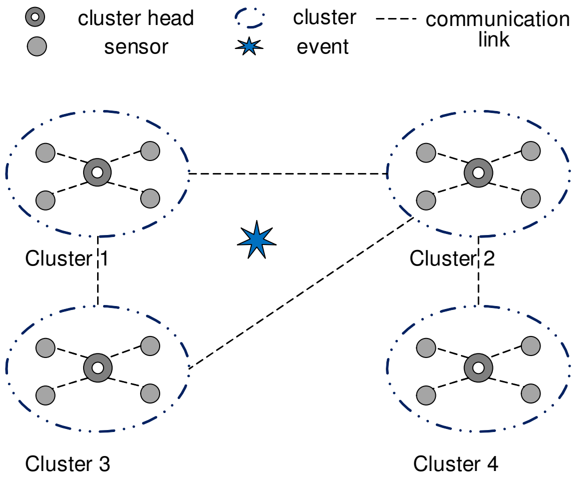


Figure 7.1: Cluster based event localization architecture ($N = 4$)

7.1 Problem Statement

Motivated by mobile acoustic event localization applications such as the PinPoint™ event localization sensor network [18, 25], we consider a localization sensor network divided into N clusters (cf. Fig. 7.1 for the case $N = 4$). Denote the number of constituent sensors of cluster i as C_i ($i = 1, 2, \dots, N$). We consider localization in D ($D \in \{1, 2, 3\}$) dimensional Euclidean space and suppose that the position of the target event is denoted as $\boldsymbol{x} \in \mathbb{R}^D$. Denote the position of the kth sensor in the ith cluster as $\boldsymbol{a}_{i,k} \in \mathbb{R}^D$. The kth sensor in the ith cluster can obtain a noisy range measurement $r_{i,k}$ of its distance with respect to a target event:

$$r_{i,k} = d_{i,k} + v_{i,k}$$

where $d_{i,k} = \parallel \boldsymbol{x} - \boldsymbol{a}_{i,k} \parallel$ denotes the actual distance between the event position and the kth sensor of the ith cluster, and $v_{i,k}$ is the Gaussian noise term.

Then the event localization problem amounts to estimating the unknown event location $\boldsymbol{x}$ using known sensor positions $\boldsymbol{a}_{i,k}$ and noisy range measurements $r_{i,k}$ ($i = 1, 2, \dots, N$, $k = 1, 2, \dots, C_i$). Still motivated by acoustic event localization applications (e.g., the PinPoint™ event localization sensor network [18, 25]), we assume that a cluster head exists in each cluster i, which can gather range measurements $r_{i,k}$ from all sensors within the cluster. In addition, a cluster head can communicate and exchange information with the cluster head of a neighboring cluster if there is a communication link between them (cf. Fig. 7.1). In this case, we also say that these two clusters can communicate. We assume that the communication pattern forms a connected network, i.e., there is a (multi-hop) path (composed of multiple communication links connected in succession) between any pair of cluster heads. For example, in Fig. 7.1, cluster 1 is able to exchange information with clusters 2 and 3 (via cluster heads); cluster 2 can exchange information with clusters 1, 3, and 4 (via cluster heads), etc. Denote $\mathcal{N}_i$ as the set of all neighboring clusters of cluster i, $\hat{\mathcal{N}_i}$ as the union of set $\mathcal{N}_i$ and cluster i itself, and N_i as the number of clusters in $\mathcal{N}_i$.

As in most existing results, we use the maximum likelihood method for event localization [30, 91]. Let $p_{i,k}(d_{i,k}(\boldsymbol{x}, \boldsymbol{a}_{i,k})|r_{i,k})$ denote the measuring probability density function (PDF) for sensor k in cluster i and assume that it is a log-concave function of unknown distance $d_{i,k}$ [106], we can write this problem using the maximum likelihood method (which is costly but efficient [11]):

$$\boldsymbol{x}^*_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{x}\in\mathbb{R}^D} \sum_{i=1}^{N}\sum_{k=1}^{C_i} \mathrm{ln} p_{i,k}(d_{i,k}(\boldsymbol{x}, \boldsymbol{a}_{i,k})|r_{i,k}). \tag{7.1}$$

7.2 Convex Relaxation

Problem (7.1) is non-convex and it is generally infeasible to find a global optimal solution [106]. So a convex relaxation is needed to convert problem (7.1) into a convex optimization problem. Following the idea of [106], we use an SDP based relaxation approach. However, it is worth noting that there are inherent differences between the problem considered here and the sensor-position estimation problem in [106] where each sensor with unknown position estimates its own position using embedded computation capability. The differences are evident from the following example. Suppose that there is only one target to localize. In the case of [106], the target will be a sensor with unknown position and it estimates its own position *alone* using a *centralized* SDP based on all information gathered from adjacent sensors, including their positions and corresponding range measurements. Whereas in our case, the target is an event without any communication or computation capability and the event position estimation process is conducted *cooperatively* in a *distributed* way among the clusters.

To facilitate the relaxation, we first define the following new variables: $y = \boldsymbol{x}^T\boldsymbol{x}$, $\epsilon_{i,k} = d^2_{i,k}$. Then we stack $\epsilon_{i,k}, k \in \{1, 2, ..., C_i\}$ into $\boldsymbol{\epsilon}_i$ and further stack $\boldsymbol{\epsilon}_i, i \in \{1, 2, ..., N\}$ into $\boldsymbol{\epsilon} \triangleq [\boldsymbol{\epsilon}_1^T, \boldsymbol{\epsilon}_2^T, ..., \boldsymbol{\epsilon}_N^T]^T$. In the same way we stack $d_{i,k}$ into $\boldsymbol{d}_i$ and $\boldsymbol{d} \triangleq [\boldsymbol{d}_1^T, \boldsymbol{d}_2^T, ..., \boldsymbol{d}_N^T]^T$. Then the cost function can be written as

$$f(\boldsymbol{d}) = -\sum_{i=1}^{N}\sum_{k=1}^{C_i} \mathrm{ln} p_{i,k}(d_{i,k}|r_{i,k}).$$

Consider the case of white zero-mean Gaussian noise, i.e., $v_{i,k} \sim \mathcal{N}(0, \sigma_{i,k}^2)$, then the above problem can be rewritten as

$$f(\boldsymbol{d}) = \sum_{i=1}^{N} \sum_{k=1}^{C_i} \sigma_{i,k}^{-2} (d_{i,k}^2 - 2d_{i,k} r_{i,k} + r_{i,k}^2) \tag{7.2}$$

Without loss of generality, we can set the standard deviation $\sigma_{i,k}$ in (7.2) to one. Now, problem (7.1) can be relaxed into the following constrained optimization problem:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{\epsilon},\boldsymbol{d},y} \quad & f(\boldsymbol{d}) \\ \text{subject to} \quad & y - 2\boldsymbol{x}^T \boldsymbol{a}_{i,k} + \| \boldsymbol{a}_{i,k} \|^2 = \epsilon_{i,k}, \quad y = \boldsymbol{x}^T \boldsymbol{x}, \\ & \epsilon_{i,k} = d_{i,k}^2, d_{i,k} \geq 0, \\ & \forall i \in \{1, 2, ..., N\}, \quad k \in \{1, 2, ..., C_i\}. \end{aligned} \tag{7.3}$$

However, in this case, the constraints of (7.3) still define a non-convex set [106]. Using Schur complements [94], the following convex relaxation can be obtained:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{\epsilon},\boldsymbol{d},y} \quad & f(\boldsymbol{d}) \\ \text{subject to} \quad & y - 2\boldsymbol{x}^T \boldsymbol{a}_{i,k} + \| \boldsymbol{a}_{i,k} \|^2 = \epsilon_{i,k}, \quad \epsilon_{i,k} \geq 0, \\ & \begin{pmatrix} 1 & d_{i,k} \\ d_{i,k} & \epsilon_{i,k} \end{pmatrix} \succeq 0, \quad d_{i,k} \geq 0, \\ & \forall i \in \{1, 2, ..., N\}, \quad k \in \{1, 2, ..., C_i\}, \\ & \begin{pmatrix} \boldsymbol{I_D} & \boldsymbol{x} \\ \boldsymbol{x}^T & y \end{pmatrix} \succeq 0, \quad y \geq 0. \end{aligned} \tag{7.4}$$

Problem (7.4) is a convex problem with inequality constraints [16]. We can rewrite the cost function as

$$f(\boldsymbol{d}, \boldsymbol{\epsilon}) = \sum_{i=1}^{N} \sum_{k=1}^{C_i} \sigma_{i,k}^{-2} (\epsilon_{i,k} - 2d_{i,k} r_{i,k} + r_{i,k}^2) \tag{7.5}$$

by enforcing a change of variables $\epsilon_{i,k} = d_{i,k}^2$ to further relax it to a semidefinite programming (SDP) problem [106]. Now, we can propose ADMM based solutions for problem (7.4).

7.3 Proposed Distributed Algorithms

7.3.1 Problem Reformulation

In distributed algorithms, neighboring nodes have to generate and exchange copies of local estimates to ensure a consistent global estimation across all nodes. In our event localization architecture, a cluster is treated as a normal node which solves a common event localization problem based on measurements obtained by sensors within the cluster. And neighboring clusters exchange intermediate computational results (through cluster heads) to guarantee that all clusters reach the same estimation value.

To better interpret our algorithms, we define a local vector

$$\boldsymbol{p}_i \triangleq (\boldsymbol{\epsilon}_i^T, \boldsymbol{d}_i^T, y_i, \boldsymbol{x}_i^T)^T \in \mathbb{R}^{2C_i+D+1}, \quad i \in \{1, 2, ..., N\},$$

which is owned by cluster i.

We let $\boldsymbol{p}$ denote the stacked vector of $\boldsymbol{p}_i$ and define a convex set

$$\mathcal{P}_i \triangleq \{\boldsymbol{p}_i | \boldsymbol{p}_i \quad \text{verifies} \quad (7.4)\}.$$

Then problem (7.4) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{p}} \quad & f(\boldsymbol{p}) \\ \text{subject to} \quad & \boldsymbol{p}_i \in \mathcal{P}_i, \quad \forall i \in \{1, 2, ..., N\}, \end{aligned} \tag{7.6}$$

where, in our situation, $f(\boldsymbol{p})$ is given as follows:

$$f(\boldsymbol{p}) = -\sum_{i=1}^{N} \sum_{k=1}^{C_i} \ln p_{i,k}(d_{i,k}|r_{i,k}) = \sum_{i=1}^{N} f_i(\boldsymbol{p}_i). \tag{7.7}$$

7.3.2 ADMM based problem formulation

From the architecture in (7.7), it is easy to see that problem (7.6) can be divided into N subproblems, which can be solved in a distributed way using ADMM by adding some constraints on $\boldsymbol{p}_i$. Next we present the basic idea based on a graph-based formulation of the communication pattern.

Using graph theory [14], the communication pattern of cluster heads can be represented by $G = \{V, E\}$, where the set V denotes the set of cluster heads, and E denotes the set of undirected edges (communication links) between clusters. We use $e_{i,j} \in E, i < j$ to denote the link (if there is) between cluster heads i and j. We use $|E|$ to represent the total number of undirected edges. In our problem formulation, each cluster is associated with a local cost function $f_i(\boldsymbol{p}_i)$, and all clusters work together to solve the problem in (7.6). Assume that the local cost function f_i is only known to cluster i, then to reach consistency (consensus) of estimated position values among all clusters, we impose a constraint $\boldsymbol{x}_i = \boldsymbol{x}_j$ if there exists an edge $e_{i,j} \in E$ between clusters i and j. Introduce a matrix $J_i = [0_{D\times(2C_i+1)}, I_D] \in \mathbb{R}^{D\times(2C_i+D+1)}$, where I_D denotes the D dimensional identity matrix, then $\boldsymbol{x}_i$ can be represented as $\boldsymbol{x}_i = J_i\boldsymbol{p}_i$. So the constraint $\boldsymbol{x}_i = \boldsymbol{x}_j$ can be represented as $J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j$.

Now we are able to rewrite problem (7.6) into a distributed ADMM form as follows:

$$\begin{aligned} \min_{\boldsymbol{p}_i,\, i\in\{1,2,\dots,N\}} \quad & \sum_{i=1}^{N} f_i(\boldsymbol{p}_i) \\ \text{subject to} \quad & J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j, \quad \forall e_{i,j} \in E, \\ & \boldsymbol{p}_i \in \mathcal{P}_i, \quad \forall i \in \{1,2,...,N\}, \end{aligned} \tag{7.8}$$

or in a more compact way:

$$\begin{aligned} \min_{\boldsymbol{p}} \quad & f(\boldsymbol{p}) \\ \text{subject to} \quad & AJ\boldsymbol{p} = 0, \quad \boldsymbol{p}_i \in \mathcal{P}_i, \quad \forall i \in \{1,2,...,N\}, \end{aligned} \tag{7.9}$$

where $\boldsymbol{p} = [\boldsymbol{p}_1^T, \boldsymbol{p}_2^T, ..., \boldsymbol{p}_N^T]^T$, $J = \text{diag}\{J_1, J_2, \dots, J_N\} \in \mathbb{R}^{ND\times(\sum_{i=1}^{N} 2C_i+D+1)}$, and $A = [a_{m,n}] \otimes I_D \in \mathbb{R}^{|E|D\times ND}$ is the edge-node incidence matrix of graph G as defined in [118], with its $|E|D$ rows corresponding to the $|E|$ communication links and the ND columns corresponding to the N agents. The symbol $\otimes$ denotes Kronecker product. The $a_{m,n}$ element is defined as

$$a_{m,n} = \begin{cases} 1 & \text{if the } m^{th} \text{ edge originates from agent } n, \\ -1 & \text{if the } m^{th} \text{ edge terminates at agent } n, \\ 0 & \text{otherwise.} \end{cases} \tag{7.10}$$

Here we define that each edge $e_{i,j}$ originates from agent i and terminates at agent j.

It can be easily verified that the incidence matrix A for Fig. 7.1 is

$$A = \begin{bmatrix} I_D & -I_D & 0_D & 0_D \\ 0_D & I_D & -I_D & 0_D \\ I_D & 0_D & -I_D & 0_D \\ 0_D & I_D & 0_D & -I_D \end{bmatrix}. \tag{7.11}$$

In this formulation, after each cluster obtains its local estimate $\boldsymbol{p}_i$, it sends the value $J_i\boldsymbol{p}_i$ (estimated event position $\boldsymbol{x}_i$) to neighboring clusters. By adding the constraint $J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j, \forall i \in \{1, 2, ..., N\}, j \in \mathcal{N}_i$ as shown in (7.8), the consistency of individual event position $J_i\boldsymbol{p}_i$ ($\boldsymbol{x}_i$) estimated across the clusters is guaranteed. Now we are in place to present our detailed algorithms to solve (7.8).

Remark 20. *Note that although a normal way to apply ADMM to consensus problems is to create auxiliary local variables (cf. [106]), we just put the constraint $J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j$ directly here. The reason that we omit the auxiliary local variables is to save storage space at each cluster, since auxiliary local variables take additional storage space. Furthermore, by adding the constraint $J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j$, we can have both a sequential and a parallel realization with convergence guaranteed, which will be detailed in the following subsection. This kind of constraint and its induced ADMM algorithm is called extended ADMM, which is discussed and applied in many recent work, e.g., [26, 82, 83, 118, 127].*

7.3.3 Proposed Algorithms

Let $\boldsymbol{\lambda}_{i,j}$ be the Lagrange multiplier relevant to the constraint $J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j$. Then the regularized augmented Lagrangian function of problem (7.8) can be reformulated as

$$\mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} f_i(\boldsymbol{p}_i) + \sum_{e_{i,j}\in E} (\boldsymbol{\lambda}_{i,j}^T(J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j) + \frac{\rho}{2} \parallel J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j \parallel^2), \tag{7.12}$$

where $\boldsymbol{\lambda}_{i,j}$ are stacked into $\boldsymbol{\lambda}_i$ for all $j \in \mathcal{N}_i$ and $\boldsymbol{\lambda}_i$ are stacked into $\boldsymbol{\lambda}$ for all $i \in \{1, 2, ..., N\}$.

Applying ADMM, we can get the following two updating recursions:

$$\boldsymbol{p}^{t+1} = \text{argmin}_{\boldsymbol{p}_i\in\mathcal{P}_i}\mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{\lambda}^t), \tag{7.13}$$

$$\boldsymbol{\lambda}_{i,j}^{t+1} = \boldsymbol{\lambda}_{i,j}^t + \rho(J_i\boldsymbol{p}_i^{t+1} - J_j\boldsymbol{p}_j^{t+1}). \tag{7.14}$$

Here, we can update $\boldsymbol{p}$ in two different ways. One way is based on the Gauss-Seidel update [37] in which clusters update in a sequential order. The other way is the Jacobian scheme in which all clusters update in parallel [99].

Gauss-Seidel update (GS-ADMM): We first consider an algorithm based on the Gauss-Seidel update. Gauss-Seidel update for distributed ADMM has been explored theoretically and proven able to converge in most cases for convex objective functions (see, e.g., [46, 52, 108]). GS-ADMM based solution for distributed event localization can be described as follows:

Algorithm 7 GS-ADMM

Each cluster initializes $\boldsymbol{p}_i^0$, $\boldsymbol{\lambda}_{i,j}^0$.
Input: $\boldsymbol{p}_i^t$, $\boldsymbol{\lambda}_{i,j}^t$
Output: $\boldsymbol{p}_i^{t+1}$, $\boldsymbol{\lambda}_{i,j}^{t+1}$

1. All clusters update their local vectors in a sequential order and send their local vectors $J_i\boldsymbol{p}_i^{t+1}$ to neighboring clusters in $\hat{\mathcal{N}}_i$ immediately, where

$$\begin{aligned}\boldsymbol{p}_i^{t+1} = \text{argmin}_{\boldsymbol{p}_i\in\mathcal{P}}(f_i(\boldsymbol{p}_i) + \sum_{j\in\hat{\mathcal{N}}_i, j\geq i}(\boldsymbol{\lambda}_{i,j}^{tT}(J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^t) + \frac{\rho}{2}\parallel J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^t\parallel^2)\\ + \sum_{j\in\hat{\mathcal{N}}_i, j<i}(\boldsymbol{\lambda}_{i,j}^{tT}(J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^{t+1}) + \frac{\rho}{2}\parallel J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^{t+1}\parallel^2)).\end{aligned} \tag{7.15}$$

   Here we also consider the effect of $J_i\boldsymbol{p}_i^t$ when updating $\boldsymbol{p}_i^{t+1}$ by adding a term $\frac{\rho}{2}\parallel J_i\boldsymbol{p}_i - J_i\boldsymbol{p}_i^t\parallel^2$. Problem (7.15) with f_i given in (7.5) is an SDP problem that can be solved by common convex toolboxes such as Yalmip [70, 106], which is used in our simulations.

2. Each cluster computes

$$\boldsymbol{\lambda}_{i,j}^{t+1} = \boldsymbol{\lambda}_{i,j}^t + \rho(J_i\boldsymbol{p}_i^{t+1} - J_j\boldsymbol{p}_j^{t+1}). \tag{7.16}$$

3. Set $t = t + 1$, and go to 1.

In GS-ADMM, all clusters update their local estimated position values in a sequential way just as some projection-based algorithms. Sequential update can be used in small-size networks. For large-scale networks, a parallel method is more appropriate. So we also propose another algorithm based on Jacobian scheme which is amendable for parallelization.

Jacobian based ADMM (J-ADMM): Algorithm J-ADMM is motivated by the work in [26], which proposed the Proximal Jacobian ADMM by adding some proximal terms when updating $\boldsymbol{p}_i$. We adopt the same idea here and prove that if the proximal terms meet some additional requirements, convergence of this algorithm can be guaranteed. The detailed procedure of J-ADMM is given as follows, with the convergence analysis detailed in the following section.

Algorithm 8 J-ADMM

```
Each cluster initializes $\boldsymbol{p}_i^0$, $\boldsymbol{\lambda}_{i,j}^0$.
Input: $\boldsymbol{p}_i^t$, $\boldsymbol{\lambda}_{i,j}^t$
Output: $\boldsymbol{p}_i^{t+1}$, $\boldsymbol{\lambda}_{i,j}^{t+1}$
    1. Each cluster updates its local vector in parallel:
       $\boldsymbol{p}_i^{t+1} = \text{argmin}_{\boldsymbol{p}_i \in \mathcal{P}} f_i(\boldsymbol{p}_i) + \sum_{j \in \hat{\mathcal{N}}_i} (\boldsymbol{\lambda}_{i,j}^{tT}(J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^t) + \frac{\rho}{2} \parallel J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^t \parallel^2) + \frac{\rho\gamma_i}{2} \parallel J_i\boldsymbol{p}_i - J_i\boldsymbol{p}_i^t \parallel^2$   (7.17)
       The last term of the above equality, i.e., $\frac{\rho\gamma_i}{2} \parallel J_i\boldsymbol{p}_i - J_i\boldsymbol{p}_i^t \parallel^2$, is the proximal term we added where
       $\gamma_i \geq 0$ is a scalar. Problem (7.17) with $f_i$ given in (7.5) is an SDP problem that can be solved by
       common convex toolboxes such as Yalmip [70, 106], which is used in our simulations.
    2. Each cluster sends its local vector $J_i\boldsymbol{p}_i^{t+1}$ to neighboring clusters in $\mathcal{N}_i$.
    3. Each cluster computes
       $\boldsymbol{\lambda}_{i,j}^{t+1} = \boldsymbol{\lambda}_{i,j}^t + \rho(J_i\boldsymbol{p}_i^{t+1} - J_j\boldsymbol{p}_j^{t+1}).$   (7.18)
    4. Set $t = t + 1$, and go to 1.
```

Remark 21. *A distinct difference between GS-ADMM and J-ADMM is the way they update $\mathbf{p}_i$. In GS-ADMM, each cluster updates its local estimated position value in a sequential way, which requires a globally predefined order. Whereas in J-ADMM, all clusters update their local estimated position values simultaneously. We remark that GS-ADMM is appropriate for small-scale sensor networks. But for large-scale networks, updating in a sequential way may be quite time-consuming and parallel methods like J-ADMM are more appropriate. So different updating methods should be chosen according to the size of networks and other practical concerns.*

In fact, if we disregard the PinPoint$^{\text{TM}}$ motivated application scenario, the proposed two algorithms can be completely distributed to each sensor by allowing sensors to have access to neighboring sensors' positions and range measurements with respect to the target event. However, we argue that this, in fact, may cost more energy since each sensor has to solve an SDP problem. In addition, the required storage overhead is larger since each sensor has to store neighboring sensors' positions and range measurements. Furthermore, consider a situation where two sensors can communicate with each other and have the same neighbors. Then the position estimation process conducted at these two sensors are the same, which leads to redundant processing of the same data. While in our clustered architecture, only cluster heads need to conduct position estimation and in fact, each sensor in the cluster can take turns to be the cluster head, which is helpful to average energy consumption. Compared with the iterative schemes, e.g., projection-based algorithms, where each sensor only has access to its own position and range measurement, our algorithms are *insensitive* to the convex hull problem. And compared with centralized SDP-based algorithms, our clustered architecture is robust to processing center failure or traffic bottleneck problems. In addition, the convex relaxation methods used at each cluster can be further improved by using recent works such as [30, 91, 92, 113, 114, 124, 126].

7.4 Convergence Analysis

In this section, we analyze the convergence properties of GS-ADMM and J-ADMM. As our algorithms are applications of distributed ADMM, the analysis benefits from many existing results on general distributed ADMM [40, 82, 118].

7.4.1 Convergence Analysis of GS-ADMM

Let $\boldsymbol{p}^k = [\boldsymbol{p}_1^{kT}, \boldsymbol{p}_2^{kT}, ..., \boldsymbol{p}_N^{kT}]^T$ and $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E}$ be the iterates generated by algorithm GS-ADMM following (7.15) and (7.16). Assume that the initial problem (7.8) admits a solution $(\boldsymbol{p}^*, \boldsymbol{\lambda}^*)$, i.e., the Lagrangian function $L(\boldsymbol{p}, \boldsymbol{\lambda}) = f(\boldsymbol{p}) + \boldsymbol{\lambda}^T AJ\boldsymbol{p}$ has a saddle point (note: not the augmented Lagrangian function), then the following theorem holds:

Theorem 14. *Let* $\bar{\boldsymbol{p}}^{t+1} = \frac{1}{t+1} \sum\limits_{k=0}^{t} \boldsymbol{p}^{k+1}$ *be the average of* $\boldsymbol{p}^k$ *up to iteration time* $t+1$*, then the followings hold for all* t*:*

(1)

$$0 \leq L(\bar{\boldsymbol{p}}^{t+1}, \boldsymbol{\lambda}^*) - L(\boldsymbol{p}^*, \boldsymbol{\lambda}^*) \leq \frac{c_0}{t+1}, \tag{7.19}$$

(2) The sequence $(\boldsymbol{p}_1^k, \boldsymbol{p}_2^k, ..., \boldsymbol{p}_N^k)$ *deduced by GS-ADMM converges to* $(\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, ..., \boldsymbol{p}_N^*)$*, i.e.,* $\lim\limits_{k \to \infty} \| \boldsymbol{p}^k - \boldsymbol{p}^* \| = 0$. *In addition, we have* $J_1 \boldsymbol{p}_1^* = J_2 \boldsymbol{p}_2^* = ... = J_N \boldsymbol{p}_N^*$.

Here

$$c_0 = \frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 + \frac{\rho}{2} (\| HJ(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \| J\boldsymbol{p}^0 - J\boldsymbol{p}^* \|^2), \tag{7.20}$$

and $H = \min\{0, A\}$ *(*$H_{i,j} = \min\{0, A_{i,j}\}$*).*

Proof: (7.19) can be obtained following a way similar to Theorem 4.4 in [118]. A detailed proof is given in Appendix C.1. To prove the second statement, recall that the objective function is

$$f(\boldsymbol{d}) = \sum_{i=1}^{N}\sum_{k=1}^{C_i} \sigma_{i,k}^{-2}(d_{i,k}^2 - 2d_{i,k}r_{i,k} + r_{i,k}^2).$$

Setting $h_{i,k} = \sigma_{i,k}^{-2}(d_{i,k}^2 - 2d_{i,k}r_{i,k} + r_{i,k}^2)$, we have $f(\boldsymbol{d}) = \sum\limits_{i=1}^{N}\sum\limits_{k=1}^{C_i} h_{i,k}$. Note that $h_{i,k}$ is a quadratic function and is strongly convex. Since the sum of strongly convex functions is still strongly convex, our objective function $f(\boldsymbol{d})$ is strongly convex. Further note that $f(\boldsymbol{p})$ is equal to $f(\boldsymbol{d})$ and the set $\mathcal{P}_i$ is convex and closed. Therefore, our problem satisfies the requirements of both strongly convex objective function and convex-and-closed constraint set in [40]. Now we proceed to prove the second statement. First, rewriting $AJ\boldsymbol{p} = 0$ in the form of $\sum\limits_{i=1}^{N}[A]_i J_i \boldsymbol{p}_i = 0$, where $[A]_i$ denotes the columns of A associated with cluster i, we can form a variational inequality $MVI(Q,U)$ similar to (5)-(6) in [40]:

$$\langle \boldsymbol{u} - \boldsymbol{u}^*, \boldsymbol{Q}(\boldsymbol{u}^*)\rangle \geq 0, \quad \forall \boldsymbol{u} \in \mathcal{U},$$

where

$$\boldsymbol{u}^* := \begin{pmatrix} \boldsymbol{p}_1^* \\ \boldsymbol{p}_2^* \\ \cdots \\ \boldsymbol{p}_N^* \\ \boldsymbol{\lambda}^* \end{pmatrix}, \quad \boldsymbol{Q}(\boldsymbol{u}^*) := \begin{pmatrix} \xi_1^* + J_1^T[A]_1^T\boldsymbol{\lambda}^* \\ \xi_2^* + J_2^T[A]_2^T\boldsymbol{\lambda}^* \\ \cdots \\ \xi_N^* + J_N^T[A]_N^T\boldsymbol{\lambda}^* \\ AJ\boldsymbol{p} \end{pmatrix},$$

$$\mathcal{U} := \prod_{i=1}^{N}\mathcal{P}_i \times \mathbb{R}^{|E|D}.$$

Then following the proof of Lemma 4.1 in [40], we can get that $(\boldsymbol{p}_1^{k+1}, ..., \boldsymbol{p}_N^{k+1}, \boldsymbol{\lambda}^{k+1})$ is a solution to $MVI(Q,U)$ if $AJ\boldsymbol{p} = 0$ and $[A]_iJ_i\boldsymbol{p}_i^k = [A]_iJ_i\boldsymbol{p}_i^{k+1}$ hold. Secondly, following the proof of Lemma 4.2 in [40], we can get the following inequality:

$$\begin{aligned}
&\langle \boldsymbol{\lambda}^* - \boldsymbol{\lambda}^k, AJ\boldsymbol{p}\rangle \geq \sum_{i=1}^{N}\omega_i \parallel \boldsymbol{p}_i^{k+1} - \boldsymbol{p}_i^* \parallel^2 + \rho \parallel AJ\boldsymbol{p}^{k+1} \parallel^2 \\
&+\rho\sum_{i=1}^{N}\langle [A]_iJ_i\boldsymbol{p}_i^{k+1} - [A]_iJ_i\boldsymbol{p}_i^*, \sum_{j=i+1}^{N}([A]_jJ_j\boldsymbol{p}_j^k - [A]_jJ_j\boldsymbol{p}_j^{k+1})\rangle \\
&-\rho\sum_{i=1}^{N}\langle [A]_iJ_i\boldsymbol{p}_i^{k+1} - [A]_iJ_i\boldsymbol{p}_i^*, \frac{1}{N_i}([A]_iJ_i\boldsymbol{p}_i^{k+1} - [A]_iJ_i\boldsymbol{p}_i^k)\rangle,
\end{aligned}$$

where $f_i(\boldsymbol{p}_i)$ is strongly convex with modulus ω_i. Thirdly, define an auxiliary block-diagonal matrix W:

$$W = \begin{pmatrix} \rho N J_1^T [A]_1^T [A]_1 J_1 & \dots & 0 & 0 \\ \dots & \ddots & \dots & \dots \\ 0 & \dots & \rho N J_N^T [A]_N^T [A]_N J_N & 0 \\ 0 & \dots & 0 & \rho^{-1} I \end{pmatrix}.$$

Then by following the idea of the proof of Lemma 4.3 in [40], the following inequality can be obtained:

$$\| \boldsymbol{u}^{k+1} - \boldsymbol{u}^* \|_W^2 \leq \| \boldsymbol{u}^k - \boldsymbol{u}^* \|_W^2 - 2\sum_{i=1}^{N} \omega_i \| \boldsymbol{p}_i^{k+1} - \boldsymbol{p}_i^* \|^2 - \rho \| AJ\boldsymbol{p}^{k+1} \|^2$$
$$+ 3N\rho \sum_{i=1}^{N} \| [A]_i J_i \boldsymbol{p}_i^{k+1} - [A]_i J_i \boldsymbol{p}_i^* \|^2,$$

where

$$\| \boldsymbol{u} \|_W^2 := \| \boldsymbol{\lambda} \|_{\rho^{-1}}^2 + \rho N (\| [A]_1 J_1 \boldsymbol{p}_1 \|^2 + \| [A]_2 J_2 \boldsymbol{p}_2 \|^2 + ... + \| [A]_N J_N \boldsymbol{p}_N \|^2).$$

Finally, when $0 < \rho < \min\limits_{1 \leq i \leq N} \{ \frac{2\omega_i}{3N \| [A]_i J_i \|^2} \}$ holds, we can get the second statement following the proof of Theorem 4.1 in [40]. ■

Remark 22. *Recall $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho AJ\boldsymbol{p}^{k+1}$, we can get*

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho AJ\boldsymbol{p}^{k+1} = \boldsymbol{\lambda}^{k-1} + \rho AJ(\boldsymbol{p}^{k+1} + \boldsymbol{p}^k) = ... = \boldsymbol{\lambda}^0 + \rho AJ \sum_{i=1}^{k+1} \boldsymbol{p}^i.$$

When $k \to \infty$, we have $\boldsymbol{\lambda}^{k+1} \to \boldsymbol{\lambda}^*$. In other words, $\boldsymbol{\lambda}^* = \boldsymbol{\lambda}^0 + \rho AJ \sum\limits_{i=1}^{\infty} \boldsymbol{p}^i$. So c_0 can be represented as:

$$c_0 = \frac{\rho}{2} \| AJ \sum_{i=1}^{\infty} \boldsymbol{p}^i \|^2 + \frac{\rho}{2} (\| HJ(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \| J\boldsymbol{p}^0 - J\boldsymbol{p}^* \|^2).$$

It is clear that c_0 will increase with an increase in ρ, so if the iteration time t is fixed, $L(\bar{\boldsymbol{p}}^{t+1}, \boldsymbol{\lambda}^*) - L(\boldsymbol{p}^*, \boldsymbol{\lambda}^*)$ will also increase with an increase in ρ. That is to say, with ρ increasing, the iteration time to reach convergence will increase, namely convergence rate will be slower. Although with an increase in ρ, the convergence rate will decrease, ρ cannot be too small. This is because if ρ is too small, the constraint $J_i\boldsymbol{p}_i = J_j\boldsymbol{p}_j$ is weak, which makes reaching consistency across clusters difficult. More detailed discussions *on selecting ρ can be found in [52].*

Directly following the statements in Theorem 14, we can obtain the following result on the convergence speed:

Theorem 15. *The convergence rate of GS-ADMM is $O(1/t)$, where t is the iteration time.*

Proof: The result can be obtained directly from the proof of Theorem 14 and is omitted. ■

7.4.2 Convergence Analysis of J-ADMM

To analyze the convergence of J-ADMM, we first define several terms: Let $\boldsymbol{p}^k = [\boldsymbol{p}_1^{kT}, \boldsymbol{p}_2^{kT}, ..., \boldsymbol{p}_N^{kT}]^T$ and $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E}$ be the results for (7.17) and (7.18) for iteration k. Augment the coefficients γ_i of proximal terms into a matrix $Q_P = \text{diag}\{\gamma_1 I_D, \gamma_2 I_D, ..., \gamma_N I_D\}$ and introduce a positive definite diagonal matrix $Q_C = \text{diag}\{N_1 I_D, N_2 I_D, ..., N_N I_D\}$, where N_i is the number of clusters in $\mathcal{N}_i$. Since Q_C and Q_P are both diagonal matrices, we can define a new diagonal matrix $\bar{Q}$ according to $\bar{Q}^T \bar{Q} = Q_C + I + Q_P$ where I is the identity matrix. It can be easily verified that $\bar{Q}$ has the following form:

$$\bar{Q} = \text{diag}\{\gamma_1' I_D, \gamma_2' I_D, ..., \gamma_N' I_D\}, \tag{7.21}$$

with $\gamma_i' > 0$ for $i = 1, 2, \ldots, N$. Assuming that the original problem (7.8) admits a solution $(\boldsymbol{p}^*, \boldsymbol{\lambda}^*)$, then we have the following theorem:

Theorem 16. *Let $\bar{\boldsymbol{p}}^{t+1} = \frac{1}{t+1} \sum_{k=0}^{t} \boldsymbol{p}^{k+1}$ be the average of $\boldsymbol{p}^k$ up to iteration time $t+1$ and denote the eigenvalues of $A^T A$ as α_i. If $\gamma_i' \geq \sqrt{\alpha_{\max}}$ is true with $\alpha_{\max} = \max\{\alpha_i\}$, then the following holds for all t: (1)*

$$0 \leq L(\bar{\boldsymbol{p}}^{t+1}, \boldsymbol{\lambda}^*) - L(\boldsymbol{p}^*, \boldsymbol{\lambda}^*) \leq \frac{c_1}{t+1}, \tag{7.22}$$

where $L(\boldsymbol{p}, \boldsymbol{\lambda}) = f(\boldsymbol{p}) + \boldsymbol{\lambda}^T A J \boldsymbol{p}$ is the Lagrangian function, and

$$c_1 = \frac{1}{2\rho} \parallel \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \parallel^2 + \frac{\rho}{2}(\parallel \bar{Q} J(\boldsymbol{p}^0 - \boldsymbol{p}^*) \parallel^2 . \tag{7.23}$$

(2) The sequence $(\boldsymbol{p}_1^k, \boldsymbol{p}_2^k, ..., \boldsymbol{p}_N^k)$ deduced by J-ADMM converges to $(\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, ..., \boldsymbol{p}_N^*)$, i.e., $\lim_{k \to \infty} \parallel \boldsymbol{p}^k - \boldsymbol{p}^* \parallel = 0$. In addition, we have $J_1 \boldsymbol{p}_1^* = J_2 \boldsymbol{p}_2^* = ... = J_N \boldsymbol{p}_N^*$.

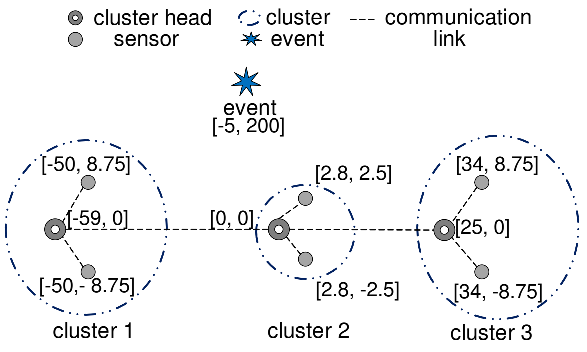


Figure 7.2: Event localization architecture used in simulations. The values in [•] denote positions (x, y coordinates) of sensors.

Proof: See Appendix C.2. ■

From Theorem 16, we can easily obtain the following results on the convergence speed:

Theorem 17. *The convergence rate of J-ADMM is $O(1/t)$, where t is the iteration time.*

Proof: The result can be obtained directly from the proof of Theorem 16 and is omitted. ■

Since c_0 and c_1 are of the same form, Remark 22 for GS-ADMM also applies to the J-ADMM case. Next, we use numerical results to evaluate the performance of GS-ADMM and J-ADMM.

7.5 Numerical Simulations

In this section, we illustrate effectiveness of the proposed approaches using comparison with existing results. A typical type of distributed algorithms for event localization is the projection-based algorithms. However, some projection-based algorithms, e.g., the DAPA algorithm in [135], is found in our simulations not appropriate for the considered case where the target event lies outside the convex hull of sensors. More specifically, we set the sensor localization architecture similar as in [18, 90], which considers a practical acoustic event localization system (see Fig. 7.2 for the detailed spatial distribution of all sensor nodes). The target event occurs at $\boldsymbol{x} = [-5; 200]$, which is far away from the nine sensors. Simulation results suggested that DAPA did not work well in this architecture, even if we set the initial values close to the target event and used the range measurements without noise, although it did work very well if the target event was set in the convex hull of sensors. In the simulation, we used the same parameters for DAPA as in [135], i.e., $\alpha_1 = ... = \alpha_9 = \frac{1}{t+2}$, $\beta_1 = ... = \beta_9 = \frac{1}{t+1}$, $b_1 = ... = b_9 = 1$, and $\xi_1 = ... = \xi_9 = 3$.

Then we compared the localization performance of the proposed algorithms GS-ADMM and J-

ADMM with two other projection-based algorithms: the PPM algorithm proposed in [56] and the PONLM algorithm proposed in [101], which gave reasonable performance in the simulations. PPM is a parallel projection method which requires a central node to average the local event location estimates obtained from all sensors in every iteration. PONLM is a sequential projection-based algorithm which solves the event localization problem by finding a point at the intersection of sensing circles. In fact, both PPM and PONLM cannot be applied to the sensor network structure in Fig. 7.2, since there exists no central node or path that connects all nine sensors in succession. We list their results here so that the performance of our algorithm can be evaluated in context. Both localization error (differences between estimated and actual target event positions) and localization consistency (differences in estimated positions between clusters) are compared under different noise standard deviations $\sigma_{i,k}$. The convergence performance is evaluated by exploring the evolution of the localization error with iteration time t.

To facilitate comparison, we first define two performance indices:

Localization Error: we use the root mean square error (RMSE) to quantify the error between estimated and true positions for every cluster or sensor, which is denoted as $\mathrm{ERR}_{\mathrm{RMSE}}$:

$$\mathrm{ERR}_{\mathrm{RMSE}} = \sqrt{\frac{\sum\limits_{j=1}^{L} \| \boldsymbol{x}_j - \boldsymbol{x}^* \|^2}{L}},$$

where L is the number of Monte Carlo trials, $\boldsymbol{x}_j$ is the estimated position in the jth Monte Carlo trial in a certain cluster or sensor, and $\boldsymbol{x}^*$ is the true position of the target event.

Localization Inconsistency: We also use the root mean square error (RMSE) to quantify the localization inconsistency (difference) in estimated event positions between N clusters, which is denoted as $\mathrm{INC}_{\mathrm{RMSE}}$:

$$\mathrm{INC}_{\mathrm{RMSE}} = \sqrt{\frac{\sum\limits_{k=1}^{L} \sum\limits_{i=1}^{N-1} \sum\limits_{j=i+1}^{N} \| \boldsymbol{x}_{i,k} - \boldsymbol{x}_{j,k} \|^2}{L}},$$

where L is the number of Monte Carlo trials, $\boldsymbol{x}_{i,k}$ is the estimated position obtained from the ith cluster in the kth Monte Carlo trial. N is the number of clusters.

7.5.1 Convergence performance

We compared the convergence performance of our sequential GS-ADMM algorithm, parallel J-ADMM algorithm, the sequential PONLM algorithm in [101], and the parallel PPM algorithm in [56]. For GS-ADMM and J-ADMM, we set $\rho = 10^{-3}$. For PPM and PONLM, we set the initial point at $[-50; 100]$ (PPM and PONLM are sensitive to initialization settings, which will be shown later). We used the range measurements without noise in this part. The simulation results are given in Fig. 7.3.

From Fig. 7.3, we can see that both GS-ADMM and J-ADMM reached an accuracy of 10^0 after about 10 iterations, while PONLM took 25 iterations and PPM took about 150 iterations. Note that sensors and clusters have to exchange local estimates in each iteration, so the required communication overhead is heavier with an increase in iteration times. The same conclusion can be drawn for energy consumption. It is worth noting that both PPM and PONLM can reach very high accuracies. However, in practical applications like gunfire localization, the accuracy of 10^0 is sufficient [18].

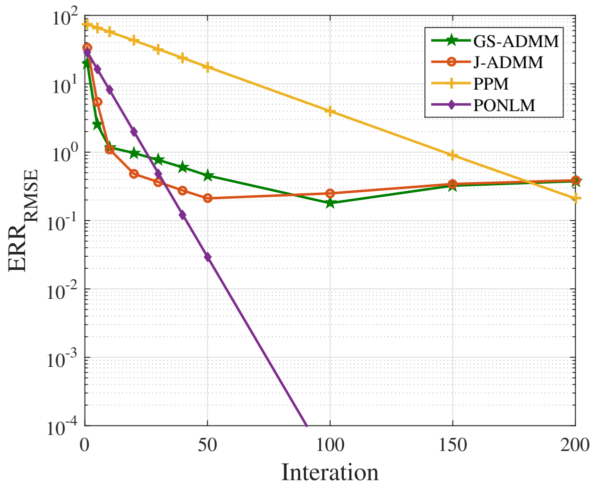


Figure 7.3: The evolution of localization error

Remark 23. *In our simulations, we used the Sedumi solver in Yalmip, whose limited precision may lead to approximate minima when solving subproblems* (7.15) *and* (7.17)*. This may also lead to a low convergence speed or even fluctuations after a certain number of iterations. In addition, SeDuMi may sometimes return the message "Run into numerical problems", which implies that it has terminated before it finds an approximate optimal solution [110]. In this situation, we can transform semi-definite inequality constraints into definite inequality constraints by introducing a constant positive definite term (e.g.,* 10^{-6}*) as indicated in [60]. However, such a transformation may bring fluctuations to the convergence process.*

7.5.2 The influence of noise level on $\mathrm{ERR}_{\mathrm{RMSE}}$

In this section, we simulated the event localization algorithms under different levels of Guassian noise standard deviation $\sigma_{i,k}$. For GS-ADMM and J-ADMM, we set $\rho = 10^{-3}$. For PPM and PONLM, we ran simulations under two cases: setting fixed initial values at $[-50; 100]$ (denote as Fix in Table 7.1) and setting random initial values in the area of 10000m $\times$ 10000m (denote as Ran in Table 7.1). The number of iterations is fixed to 50 for GS-ADMM, J-ADMM, PONLM, and 200 for PPM. All simulation results are summarized in Table 7.1 and Fig. 7.4. Each data point in Table 7.1 is an average of 100 Monte Carlo trials.

Table 7.1: $\mathrm{ERR}_{\mathrm{RMSE}}$ of GS-ADMM, J-ADMM, PPM, and PONLM under different measurement noise

$\sigma_{i,k}$	**GS-ADMM**			**J-ADMM**			**PPM**		**PONLM**	
	CL_1	CL_2	CL_3	CL_1	CL_2	CL_3	Fix	Ran	Fix	Ran
0.00	0.3693	0.4848	0.5128	0.1146	0.2223	0.2964	0.2100	273.19	0.0338	304.52
0.01	0.5417	0.5443	0.5753	0.3546	0.3911	0.4385	0.2106	285.65	0.0498	269.86
0.02	0.5453	0.5862	0.5992	0.2766	0.3266	0.3779	0.2145	282.83	0.0865	307.95
0.05	0.6055	0.6723	0.7188	0.4987	0.5261	0.5724	0.2265	268.31	0.1895	278.84
0.10	1.0564	1.0942	1.1440	1.0562	1.0589	1.1017	0.2832	288.41	0.4019	243.67

From Table 7.1, we can see that both PPM and PONLM reached high localization accuracies under fixed initial values. However, their performance deteriorated significantly when random initial values were used. Therefore PPM and PONLM are sensitive to initial value settings. If the target event lies outside the convex hull of sensors, the convergent values of PPM and PONLM may be far away from the true event position. GS-ADMM and J-ADMM can avoid the convex hull problem, so every estimate lay close to the true event position.

Fig. 7.4 visualizes the estimated event locations. Fig. 7.4 (a) and (b) show the localization results of the proposed algorithms GS-ADMM and J-ADMM respectively from 100 Monte Carlo trials with $\rho = 10^{-3}$. Fig. 7.4 (c) and (d) show the results of PPM and PONLM respectively where the initial positions are chosen randomly. It is clear that both GS-ADMM and J-ADMM performed better than PPM and PONLM when the initial values are randomly chosen.

7.5.3 The influence of noise level on $\mathrm{INC}_{\mathrm{RMSE}}$

Setting $\rho = 10^{-3}$, we also evaluated the influence of noise level on $\mathrm{INC}_{\mathrm{RMSE}}$ of our proposed algorithms. The results are summarized in Fig. 7.5.

Fig. 7.5 indicates that the proposed GS-ADMM and J-ADMM have small localization inconsistency ($\mathrm{INC}_{\mathrm{RMSE}}$) under different noise strength. In other words, our proposed algorithms GS-ADMM and J-ADMM

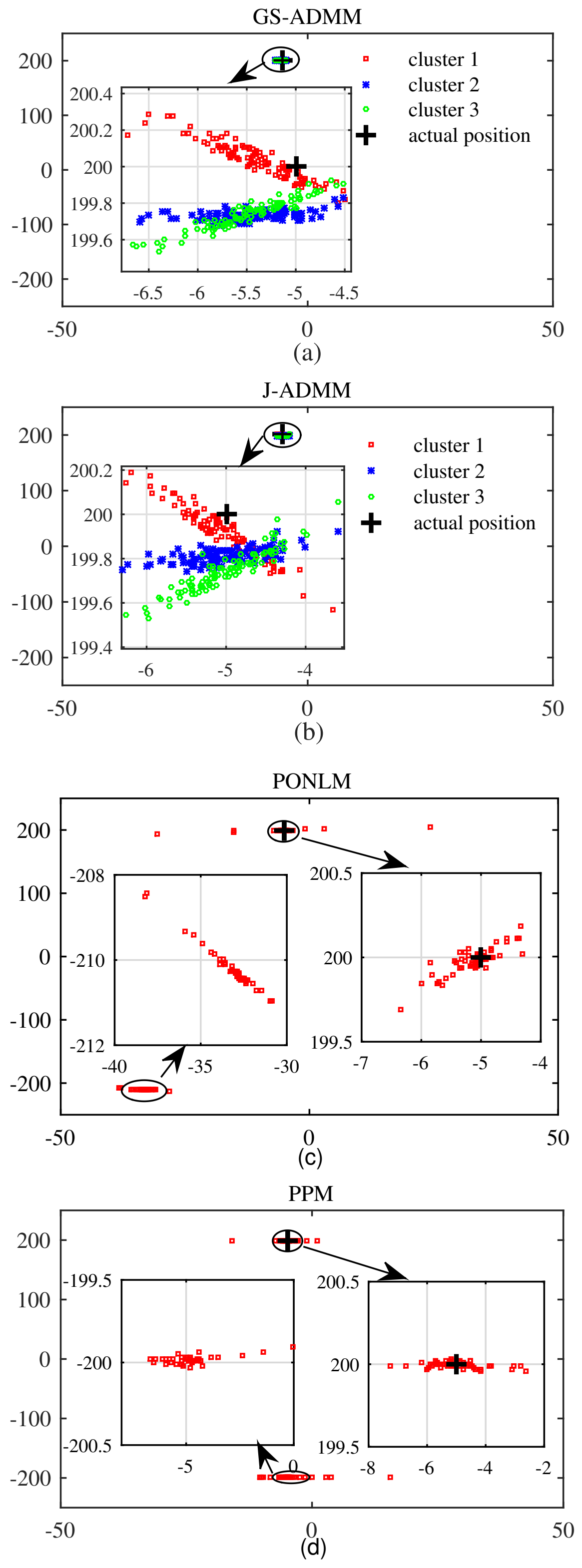


Figure 7.4: The distribution of estimated event location, $\sigma = 0.05$. (a) GS-ADMM; (b) J-ADMM; (c) PONLM; (d) PPM.

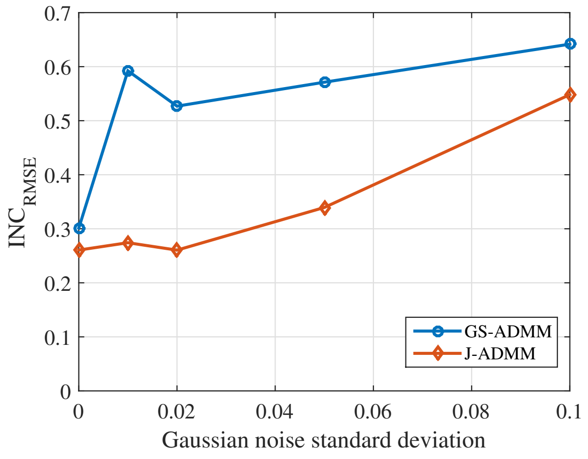


Figure 7.5: The influence of measurement noise on localization inconsistency

can achieve good consistency across clusters even under large noise standard deviations. As indicated before, consistency is of crucial importance in many applications.

7.6 Summaries

We proposed two ADMM based distributed event localization algorithms GS-ADMM and J-ADMM that do not require the target event to be within the convex hull of the deployed sensors. Convergence properties of the algorithms are analyzed theoretically. Numerical simulations showed that the proposed algorithms are robust to measurement noises and insensitive to convex hull problem compared with existing projection-based algorithms.

In addition, we would like to thank Dr. Andrea Simonetto for providing Matlab codes for his paper [106].

Chapter 8

Conclusions

In this dissertation, we addressed decentralized optimization and its application to the event localization problem. We first presented a privacy-preserving decentralized optimization approach by proposing a new ADMM and leveraging partially homomorphic cryptography. By incorporating Paillier cryptosystem into the newly proposed decentralized ADMM, our approach provides guarantee for privacy preservation without compromising the solution in the absence of any aggregator or third party. We then presented another privacy-preserving decentralized optimization algorithm based on the integration of partially homomorphic cryptography with subgradient method. Given that encryption-based algorithms unavoidably suffer from significant computational and communication overhead, we also presented a privacy-preserving solution to decentralized optimization using function decomposition. Theoretical analysis confirms that an honest-but-curious adversary cannot infer the information of neighboring agents even by recording and analyzing the information exchanged in multiple iterations. In sharp contrast to differential-privacy based approaches which protect privacy through injecting noise and are subject to a fundamental trade-off between privacy and accuracy, all algorithms can preserve privacy without sacrificing accuracy. Numerical and experimental results are given to confirm the effectiveness and efficiency of the proposed algorithms.

We also addressed the application of decentralized optimization to the event localization problem. We first proposed a completely decentralized parallel algorithm which solves the non-convex and non-smooth event localization problem directly without using convex relaxation. Simulation results confirm that this algorithm has better localization accuracy compared with other projection based algorithms when the target event is in the convex hull of sensors. When the target event is outside the convex hull of sensors, this algorithm has a higher probability to converge to the right target event position than existing results. We then proposed another two ADMM based distributed event localization algorithms GS-ADMM and J-ADMM that do not require the target event to be within the convex hull of the deployed sensors. Convergence properties of the algorithms are analyzed theoretically. Numerical simulations showed that the proposed algorithms are robust to measurement noises and insensitive to convex hull problem compared with existing projection-based algorithms.

Appendices

Appendix A Proofs of Theorems in Chapter 3

A.1 Proof of Theorem 1

The key idea to prove Theorem 1 is to show that Algorithm 1 converges to the saddle point of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$. To achieve this goal, we introduce a variational inequality $MVI(Q, U)$ first and prove that the solution of $MVI(Q, U)$ is also the saddle point of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$ (which is formulated as Lemma 9). Then we introduce a sufficient condition for solving $MVI(Q, U)$ in Lemma 10. After the two steps, what is left is to prove that the iterates of Algorithm 1 satisfy the condition in Lemma 10 when $k \to \infty$, i.e., Algorithm 1 converges to the solution of $MVI(Q, U)$ (Theorem 18 and Theorem 19).

We form a variational inequality $MVI(Q, U)$ similar to (5)-(6) in [40] first:

$$\langle \boldsymbol{u} - \boldsymbol{u}^*, \boldsymbol{Q}(\boldsymbol{u}^*) \rangle \geq \boldsymbol{0}, \quad \forall \boldsymbol{u}, \tag{1}$$

where

$$\boldsymbol{u}^* := \begin{pmatrix} \boldsymbol{x}_1^* \\ \boldsymbol{x}_2^* \\ \vdots \\ \boldsymbol{x}_N^* \\ \boldsymbol{\lambda}^* \end{pmatrix}, \quad \boldsymbol{Q}(\boldsymbol{u}^*) := \begin{pmatrix} \xi_1^* + [A]_1^T \boldsymbol{\lambda}^* \\ \xi_2^* + [A]_2^T \boldsymbol{\lambda}^* \\ \vdots \\ \xi_N^* + [A]_N^T \boldsymbol{\lambda}^* \\ A\boldsymbol{x}^* \end{pmatrix}, \tag{2}$$

$$\xi_i^* \in \partial f_i(\boldsymbol{x}_i^*), \forall i \in \{1, 2, ..., N\}.$$

In (2), $[A]_i$ denotes the columns of matrix A that are associated with agent i. By recalling the first-order necessary and sufficient condition for convex programming [40], it is easy to see that solving problem (3.6) amounts to solving the above $MVI(Q, U)$ [40]. Denote the solution set of $MVI(Q, U)$ as $\mathcal{U}^*$. Since f_i is convex, $\partial f_i(\boldsymbol{x}_i)$ is monotone, the $MVI(Q, U)$ is solvable and $\mathcal{U}^*$ is nonempty [40].

Next, we introduce several lemmas and theorems that contribute to the proof of Theorem 1.

Lemma 9. *Each $\boldsymbol{u}^* = (\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ in $\mathcal{U}^*$ is also the saddle point of the Lagrangian function $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$.*

Proof: The results can be obtained from Part 2.1 in [47] directly. ■

Lemma 10. *If* $A\boldsymbol{x}^{k+1} = \mathbf{0}$ *and* $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$ *hold, then* $(\boldsymbol{x}_1^{k+1}, \boldsymbol{x}_2^{k+1}, ..., \boldsymbol{x}_N^{k+1}, \boldsymbol{\lambda}^{k+1})$ *is a solution to* $MVI(Q, U)$.

Proof: Using the definition of matrix A and the update rule of $\boldsymbol{\lambda}^{k+1}$ in (3.11), we can see that the assumption $A\boldsymbol{x}^{k+1} = \mathbf{0}$ implies $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k$ and $\boldsymbol{x}_1^{k+1} = \boldsymbol{x}_2^{k+1} = ... = \boldsymbol{x}_N^{k+1}$.

On the other hand, we know that $\boldsymbol{x}_i^{k+1}$ is the optimizer of (3.13). By using the first-order optimality condition, we get

$$(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T(\xi_i^{k+1} + \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^k + \rho_{i,j}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k)) + \gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)) \geq 0. \tag{3}$$

where $\xi_i^{k+1} \in \partial f_i(\boldsymbol{x}_i^{k+1})$. Then based on the assumption $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$, the fact $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k$, and the definition of matrix A, we have $(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T(\xi_i^{k+1} + [A]_i^T\boldsymbol{\lambda}^{k+1}) \geq 0$. Therefore, $(\boldsymbol{x}_1^{k+1}, \boldsymbol{x}_2^{k+1}, ..., \boldsymbol{x}_N^{k+1}, \boldsymbol{\lambda}^{k+1})$ is a solution to $MVI(Q, U)$. ■

Lemma 10 provides a sufficient condition for solving $MVI(Q, U)$. According to Lemma 9, we know that the solution to $MVI(Q, U)$ is also the saddle point of the Lagrangian function. Next, we prove that the iterates in Algorithm 1 satisfy $\lim_{k\to\infty} A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\lim_{k\to\infty} \boldsymbol{x}^{k+1} - \boldsymbol{x}^k = \mathbf{0}$, i.e., Algorithm 1 converges to the solution to $MVI(Q, U)$. To achieve this goal, we first establish the relationship (4) about iterates k and $k+1$ in Theorem 18, whose proof is mainly based on convex properties. Then based on the relationship, we further prove $\lim_{k\to\infty} A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\lim_{k\to\infty} \boldsymbol{x}^{k+1} - \boldsymbol{x}^k = \mathbf{0}$ in Theorem 19.

Theorem 18. *Let* $\boldsymbol{\rho}^k$ *satisfy Condition A,* $\bar{Q} \triangleq Q_P + Q_C^k$ *satisfy Condition B, and* $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ *be the saddle point of the Lagrangian function* $L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T A\boldsymbol{x}$*, then we have*

$$\begin{aligned}&\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|_{(\boldsymbol{\rho}^{k+1})^{-1}}^2 + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|_{\bar{Q}}^2 \leq \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|_{(\boldsymbol{\rho}^k)^{-1}}^2 + \| \boldsymbol{x}^k - \boldsymbol{x}^* \|_{\bar{Q}}^2 \\ &- (\| A\boldsymbol{x}^{k+1} \|_{\boldsymbol{\rho}^k}^2 + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|_{-A^T\boldsymbol{\rho}^k A + \bar{Q}}^2) + \| A\boldsymbol{x}^{k+1} \|_{\boldsymbol{\rho}^{k+1}}^2 - \| A\boldsymbol{x}^k \|_{\boldsymbol{\rho}^k}^2 .\end{aligned} \tag{4}$$

To prove Theorem 18, we first introduce two lemmas:

Lemma 11. *Let* $\boldsymbol{x}^k = [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, ..., \boldsymbol{x}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j}\in E}$ *be the intermediate results of iteration* k *in Algorithm 1, then the following inequality holds for all* k*:*

$$f(\boldsymbol{x}) - f(\boldsymbol{x}^{k+1}) + (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T A^T\boldsymbol{\lambda}^k + (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq 0, \tag{5}$$

where $\bar{Q} \triangleq Q_P + Q_C^k$.

Proof: The proof follows from [130]. For completeness, we sketch the proof here. Denote by g_i the function

$$g_i^k(\boldsymbol{x}_i) = \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^{kT}\boldsymbol{x}_i + \frac{\rho_{i,j}^k}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_j^k \parallel^2) + \frac{\gamma_i}{2} \parallel \boldsymbol{x}_i - \boldsymbol{x}_i^k \parallel^2 . \tag{6}$$

Using $\xi_i^{k+1} \in \partial f_i(\boldsymbol{x}_i^{k+1})$, we can get $\xi_i^{k+1} + \nabla g_i(\boldsymbol{x}_i^{k+1}) = \mathbf{0}$ and $(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T[\xi_i^{k+1} + \nabla g_i(\boldsymbol{x}_i^{k+1})] = 0$ based on the fact that $\boldsymbol{x}_i^{k+1}$ is the optimizer of $g_i^k + f_i$. On the other hand, as f_i is convex, the following relationship holds:

$$f_i(\boldsymbol{x}_i) \geq f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\xi_i^{k+1}.$$

Then we can get $f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\nabla g_i(\boldsymbol{x}_i^{k+1}) \geq 0$.

Substituting $\nabla g_i(\boldsymbol{x}_i^{k+1})$ with (6), we obtain

$$f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \cdot (\sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^k + \rho_{i,j}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k)) + \gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)) \geq 0.$$

Noting $\boldsymbol{\lambda}_{i,i} = \mathbf{0}$ and $\boldsymbol{\lambda}_{i,j} = -\boldsymbol{\lambda}_{j,i}$, based on the definition of matrices A and $\boldsymbol{\rho}$, we can rewrite the above inequality as

$$f_i(\boldsymbol{x}_i) - f_i(\boldsymbol{x}_i^{k+1}) + (\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \cdot ([A]_i^T\boldsymbol{\lambda}^k + \sum_{j\in\mathcal{N}_i}\rho_{i,j}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k) + \gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k)) \geq 0. \tag{7}$$

Summing both sides of (7) over $i = 1, 2, \ldots, N$, and using

$$\begin{aligned}
&\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T[A]_i^T\boldsymbol{\lambda}^k = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T A^T\boldsymbol{\lambda}^k,\\
&\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \sum_{j\in\mathcal{N}_i}\rho_{i,j}^k\boldsymbol{x}_i^{k+1} = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T Q_C^k\boldsymbol{x}^{k+1},\\
&\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T \sum_{j\in\mathcal{N}_i}\rho_{i,j}^k\boldsymbol{x}_j^k = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T(-A^T\boldsymbol{\rho}^k A + Q_C^k)\boldsymbol{x}^k,\\
&\sum_{i=1}^N(\boldsymbol{x}_i - \boldsymbol{x}_i^{k+1})^T\gamma_i(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) = (\boldsymbol{x} - \boldsymbol{x}^{k+1})^T Q_P(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k),
\end{aligned}$$

we can get the lemma. ■

Lemma 12. *Let* $\boldsymbol{x}^k = [\boldsymbol{x}_1^{kT}, \boldsymbol{x}_2^{kT}, ..., \boldsymbol{x}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij,e_{i,j}\in E}$ *be the intermediate results of iteration*

k in Algorithm 1, then the following equality holds for all k:

$$\begin{aligned}
&-(\boldsymbol{x}^{k+1})^T A^T(\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*)-(\boldsymbol{x}^{k+1})^T A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k+(\boldsymbol{x}^*-\boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)\\
&=\frac{1}{2}(\|\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*\|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}-\|\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^*\|^2_{(\boldsymbol{\rho}^{k+1})^{-1}})+\frac{1}{2}\|\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k\|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}-\frac{1}{2}\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\|^2_{\bar{Q}}\\
&-\frac{1}{2}\|A\boldsymbol{x}^{k+1}\|^2_{\boldsymbol{\rho}^k}-\frac{1}{2}\|A\boldsymbol{x}^k\|^2_{\boldsymbol{\rho}^k}+\frac{1}{2}\|A(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)\|^2_{\boldsymbol{\rho}^k}-\frac{1}{2}(\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\|^2_{\bar{Q}}-\|\boldsymbol{x}^k-\boldsymbol{x}^*\|^2_{\bar{Q}}).
\end{aligned}\tag{8}$$

Proof: For a scalar a, we have $a^T=a$. Recall $\boldsymbol{\lambda}^{k+1}=\boldsymbol{\lambda}^k+\boldsymbol{\rho}^{k+1}A\boldsymbol{x}^{k+1}$ and notice that $\boldsymbol{\rho}^{k+1}$ is a positive definite diagonal matrix, we can get

$$(\boldsymbol{x}^{k+1})^T A^T(\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*)=(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k)^T(\boldsymbol{\rho}^{k+1})^{-1}(\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*).\tag{9}$$

On the other hand, since $(\boldsymbol{x}^*,\boldsymbol{\lambda}^*)$ is the saddle point of the Lagrangian function (3.15), we can get $A\boldsymbol{x}^*=\mathbf{0}$ [118]. Moreover, the following equalities can be established by using algebraic manipulations:

$$(\boldsymbol{x}^{k+1}-\boldsymbol{x}^*)^T\bar{Q}(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)=\frac{1}{2}\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^k\|^2_{\bar{Q}}+\frac{1}{2}(\|\boldsymbol{x}^{k+1}-\boldsymbol{x}^*\|^2_{\bar{Q}}-\|\boldsymbol{x}^k-\boldsymbol{x}^*\|^2_{\bar{Q}}),\tag{10}$$

$$-\boldsymbol{x}^{(k+1)T}A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k=\frac{1}{2}\|A(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)\|^2_{\boldsymbol{\rho}^k}-\frac{1}{2}\|A\boldsymbol{x}^{k+1}\|^2_{\boldsymbol{\rho}^k}-\frac{1}{2}\|A\boldsymbol{x}^k\|^2_{\boldsymbol{\rho}^k},\tag{11}$$

$$\begin{aligned}
(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k)^T(\boldsymbol{\rho}^{k+1})^{-1}(\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*)&=\frac{1}{2}(\|\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^*\|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}-\|\boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*\|^2_{(\boldsymbol{\rho}^{k+1})^{-1}})\\
&-\frac{1}{2}\|\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k\|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}.
\end{aligned}\tag{12}$$

Then we can obtain (8) by plugging equalities (9)-(12) into the left hand side of (8). ■

Now we can proceed to prove Theorem 18. By setting $\boldsymbol{x}=\boldsymbol{x}^*$ in (5), we can get

$$f(\boldsymbol{x}^*)-f(\boldsymbol{x}^{k+1})+(\boldsymbol{x}^*-\boldsymbol{x}^{k+1})^T A^T\boldsymbol{\lambda}^k+(\boldsymbol{x}^*-\boldsymbol{x}^{k+1})^T A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k+(\boldsymbol{x}^*-\boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)\geq 0$$

Recalling $A\boldsymbol{x}^*=\mathbf{0}$, the above inequality can be rewritten as

$$f(\boldsymbol{x}^*)-f(\boldsymbol{x}^{k+1})-\boldsymbol{x}^{(k+1)T}A^T\boldsymbol{\lambda}^k-\boldsymbol{x}^{(k+1)T}A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k+(\boldsymbol{x}^*-\boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1}-\boldsymbol{x}^k)\geq 0.\tag{13}$$

Now adding and subtracting the term $\boldsymbol{\lambda}^{*T}A\boldsymbol{x}^{k+1}$ from the left hand side of (13) gives

$$\begin{aligned} &f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T}A\boldsymbol{x}^{k+1} - \boldsymbol{x}^{(k+1)T}A^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*) \\ &\qquad - \boldsymbol{x}^{(k+1)T}A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq 0. \end{aligned} \tag{14}$$

Using $L(\boldsymbol{x}, \boldsymbol{\lambda}^*) - L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \geq 0$ and $A\boldsymbol{x}^* = \mathbf{0}$, we have

$$\begin{aligned} &- \boldsymbol{x}^{(k+1)T}A^T(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^*) - \boldsymbol{x}^{(k+1)T}A^T\boldsymbol{\rho}^k A\boldsymbol{x}^k + (\boldsymbol{x}^* - \boldsymbol{x}^{k+1})^T\bar{Q}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \\ &\geq f(\boldsymbol{x}^{k+1}) + \boldsymbol{\lambda}^{*T}A\boldsymbol{x}^{k+1} - f(\boldsymbol{x}^*) \geq 0. \end{aligned}$$

Now by plugging (8) into the left hand side of the above inequality, we can obtain

$$\begin{aligned} &\tfrac{1}{2}(\parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}}) + \tfrac{1}{2} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \tfrac{1}{2} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel^2_{\bar{Q}} \\ &- \tfrac{1}{2} \parallel A\boldsymbol{x}^{k+1} \parallel^2_{\boldsymbol{\rho}^k} - \tfrac{1}{2} \parallel A\boldsymbol{x}^k \parallel^2_{\boldsymbol{\rho}^k} + \tfrac{1}{2} \parallel A\boldsymbol{x}^{k+1} - A\boldsymbol{x}^k \parallel^2_{\boldsymbol{\rho}^k} - \tfrac{1}{2} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2_{\bar{Q}} + \tfrac{1}{2} \parallel \boldsymbol{x}^k - \boldsymbol{x}^* \parallel^2_{\bar{Q}} \geq 0. \end{aligned}$$

Noting $\parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}} = \parallel A\boldsymbol{x}^{k+1} \parallel^2_{\boldsymbol{\rho}^{k+1}}$, the above inequality can be rewritten as

$$\begin{aligned} &\parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2_{\bar{Q}} \leq \parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \parallel \boldsymbol{x}^k - \boldsymbol{x}^* \parallel^2_{\bar{Q}} \\ &\quad -(\parallel A\boldsymbol{x}^{k+1} \parallel^2_{\boldsymbol{\rho}^k} + \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel^2_{-A^T\boldsymbol{\rho}^k A + \bar{Q}}) + \parallel A\boldsymbol{x}^{k+1} \parallel^2_{\boldsymbol{\rho}^{k+1}} - \parallel A\boldsymbol{x}^k \parallel^2_{\boldsymbol{\rho}^k}. \end{aligned} \tag{15}$$

Recall that from Condition A, $\boldsymbol{\rho}^{k+1} \succeq \boldsymbol{\rho}^k$ and $\boldsymbol{\rho}^k$ $(k = 1, 2, ...)$ are positive definite diagonal matrices. So we have $(\boldsymbol{\rho}^{k+1})^{-1} \preceq (\boldsymbol{\rho}^k)^{-1}$ [58], and consequently $\parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^{k+1})^{-1}} \preceq \parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^k)^{-1}}$, which proves Theorem 18. ■

Theorem 18 established the relationship between iterates k and $k + 1$ in Algorithm 1. Based on this relationship, we can have the following theorem which shows that Algorithm 1 converges to the solution to $MVI(Q, U)$.

Theorem 19. *Let* $\boldsymbol{u}^k = (\boldsymbol{x}^k, \boldsymbol{\lambda}^k)$ *be the sequence generated by Algorithm 1, then we have*

$$\lim_{k\to\infty}(\parallel A\boldsymbol{x}^{k+1} \parallel^2_{\boldsymbol{\rho}^k} + \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel^2_{-A^T\boldsymbol{\rho}^k A + \bar{Q}}) = 0. \tag{16}$$

Proof: Let $\alpha^k = \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^k)^{-1}} + \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}}$. According to Theorem 18, we have

$$\begin{aligned}
\alpha^{k+1} &\leq \alpha^k + \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}} - \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k} - (\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T\boldsymbol{\rho}^k A + \bar{Q}}) \\
&\leq \dots \\
&\leq \alpha^0 + \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^{k+1}} - \| A\boldsymbol{x}^0 \|^2_{\boldsymbol{\rho}^0} - \sum_{i=0}^{k} (\| A\boldsymbol{x}^{i+1} \|^2_{\boldsymbol{\rho}^i} + \| \boldsymbol{x}^{i+1} - \boldsymbol{x}^i \|^2_{-A^T\boldsymbol{\rho}^i A + \bar{Q}}) \\
&\leq \alpha^0 + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T\boldsymbol{\rho}^{k+1}A} - \sum_{i=0}^{k} (\| A\boldsymbol{x}^{i+1} \|^2_{\boldsymbol{\rho}^i} + \| \boldsymbol{x}^{i+1} - \boldsymbol{x}^i \|^2_{-A^T\boldsymbol{\rho}^i A + \bar{Q}}).
\end{aligned} \tag{17}$$

The last inequality comes from the fact that $A\boldsymbol{x}^* = \mathbf{0}$ and $\| A\boldsymbol{x}^{k+1} - A\boldsymbol{x}^* \|^2_{\boldsymbol{\rho}^{k+1}}$ can be written as $\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T\boldsymbol{\rho}^{k+1}A}$. Recall that $\boldsymbol{\rho}^0 \preceq \boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1} \preceq \bar{\boldsymbol{\rho}}$ holds and $\bar{Q} - A^T\bar{\boldsymbol{\rho}}A$ is positive definite. Moving the term $\| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T\boldsymbol{\rho}^{k+1}A}$ to the left hand side of the above inequality, we have

$$\begin{aligned}
&\lim_{k\to\infty} (\alpha^{k+1} - \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{A^T\boldsymbol{\rho}^{k+1}A}) \\
= &\lim_{k\to\infty} (\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q} - A^T\boldsymbol{\rho}^{k+1}A}) \geq 0
\end{aligned} \tag{18}$$

Since α^0 is positive and bounded and $\| A\boldsymbol{x}^{i+1} \|^2_{\boldsymbol{\rho}^i} + \| \boldsymbol{x}^{i+1} - \boldsymbol{x}^i \|^2_{-A^T\boldsymbol{\rho}^i A + \bar{Q}}$ is nonnegative, following Theorem 3 in [45], we have

$$\lim_{k\to\infty} (\| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} + \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{-A^T\boldsymbol{\rho}^k A + \bar{Q}}) = 0. \tag{19}$$

■

Given that $\boldsymbol{\rho}^k$ satisfies Condition A and $\bar{Q}$ satisfies Condition B, we have that both $-A^T\boldsymbol{\rho}^k A + \bar{Q}$ and $\boldsymbol{\rho}^k$ are positive symmetric definite. Then according to Theorem 19, we have $A\boldsymbol{x}^{k+1} = \mathbf{0}$ and $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k$ when $k \to \infty$.

Therefore, based on Lemma 10, we have that $(\boldsymbol{x}^{k+1}, \boldsymbol{\lambda}^{k+1})$ in Algorithm 1 converges to a solution to $MVI(Q, U)$, i.e., a saddle point of the Lagrangian function (3.15) according to Lemma 9. Since the objective function is convex, we can conclude Theorem 1 [118]. ■

A.2 Proof of Theorem 2

Now we prove that the convergence rate of Algorithm 1 is $O(1/t)$. By plugging (8) into the left hand side of (14), we can obtain

$$f(\boldsymbol{x}^*) - f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A\boldsymbol{x}^{k+1} - \frac{1}{2}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}) - \frac{1}{2} \| A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k}$$
$$+\frac{1}{2} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}} + \frac{1}{2} \| A\boldsymbol{x}^{k+1} - A\boldsymbol{x}^k \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \|^2_{\bar{Q}}$$
$$+\frac{1}{2} \| \boldsymbol{x}^k - \boldsymbol{x}^* \|^2_{\bar{Q}} - \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q}} \geq 0.$$

Summing both sides of the above inequality over $k = 0, 1, ..., t$, we have

$$(t+1)f(\boldsymbol{x}^*) - \sum_{k=0}^{t} f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A \sum_{k=0}^{t} \boldsymbol{x}^{k+1} - \frac{1}{2} \| \boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{t+1})^{-1}} + \frac{1}{2} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^1)^{-1}}$$
$$-\sum_{k=1}^{t} \frac{1}{2}(\| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^k)^{-1}} - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^{k+1})^{-1}}) + \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \sum_{k=0}^{t} \frac{1}{2} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k}$$
$$-\frac{1}{2} \| A\boldsymbol{x}^0 \|^2_{\boldsymbol{\rho}^0} - \frac{1}{2} \| \boldsymbol{x}^{t+1} - \boldsymbol{x}^* \|^2_{\bar{Q}} + \frac{1}{2} \| \boldsymbol{x}^0 - \boldsymbol{x}^* \|^2_{\bar{Q}} - \sum_{k=0}^{t} \frac{1}{2} \| \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \|^2_{\bar{Q} - A^T \boldsymbol{\rho}^k A} \geq 0.$$

Following the above inequality, It is easy to obtain

$$(t+1)f(\boldsymbol{x}^*) - \sum_{k=0}^{t} f(\boldsymbol{x}^{k+1}) - \boldsymbol{\lambda}^{*T} A \sum_{k=0}^{t} \boldsymbol{x}^{k+1} + \frac{1}{2} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2_{(\boldsymbol{\rho}^1)^{-1}} + \frac{1}{2} \| \boldsymbol{x}^0 - \boldsymbol{x}^* \|^2_{\bar{Q}}$$
$$+\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^t} \geq 0.$$

Recall that in (19), we have proven $\lim_{k\to\infty} \| A\boldsymbol{x}^{k+1} \|^2_{\boldsymbol{\rho}^k} = 0$. Then the relationship $\boldsymbol{\rho}^0 \preceq \boldsymbol{\rho}^k \preceq \boldsymbol{\rho}^{k+1} \preceq \bar{\boldsymbol{\rho}}$ implies

$$\lim_{t\to\infty} (\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^t}) = 0.$$

Therefore, there exists some constant c such that

$$\frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^{t+1}} - \frac{1}{2} \| A\boldsymbol{x}^{t+1} \|^2_{\boldsymbol{\rho}^t} \leq c.$$

On the other hand, as our function is convex, we have $\sum_{k=0}^{t} f(\boldsymbol{x}^{k+1}) \geq (t+1)f(\bar{\boldsymbol{x}}^{t+1})$ where

$\bar{\boldsymbol{x}}^{t+1} = \frac{1}{t+1}\sum_{k=0}^{t} \boldsymbol{x}^{k+1}$. Therefore, we have

$$(t+1)f(\boldsymbol{x}^*) - (t+1)f(\bar{\boldsymbol{x}}^{t+1}) - (t+1)\boldsymbol{\lambda}^{*T} A\bar{\boldsymbol{x}}^{t+1} + \frac{1}{2} \parallel \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^1)^{-1}} + \frac{1}{2} \parallel \boldsymbol{x}^0 - \boldsymbol{x}^* \parallel^2_{\bar{Q}} + c \geq 0.$$

By dividing both sides by $-(t+1)$, we can obtain

$$f(\bar{\boldsymbol{x}}^{t+1}) + \boldsymbol{\lambda}^{*T} A\bar{\boldsymbol{x}}^{t+1} - f(\boldsymbol{x}^*) \leq \frac{1}{t+1}(\frac{1}{2} \parallel \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \parallel^2_{(\boldsymbol{\rho}^1)^{-1}} + \frac{1}{2} \parallel \boldsymbol{x}^0 - \boldsymbol{x}^* \parallel^2_{\bar{Q}} + c).$$

Combining the above relationship with the Lagrangian function (3.15), we can conclude Theorem 2.

■

Appendix B Proofs of Theorems in Chapter 5

B.1 Proof of Lemma 2

According to the update rules in (5.6) and (5.7) , we have

$$\nabla h_i^{k+1}(\boldsymbol{x}_i^{k+1}) + \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{i,j}^k + \rho(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^k)) + \gamma_i\rho(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k) = \mathbf{0} \tag{20}$$

for all $i = 1, 2, \dots, 2N$.

Rewriting (20) in a compact form, we have

$$\nabla h^{k+1}(\boldsymbol{x}^{k+1}) + A^T\boldsymbol{\lambda}^k + \rho\bar{D}\boldsymbol{x}^{k+1} - \rho(\bar{D} - A^TA)\boldsymbol{x}^k + \rho U(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) = \mathbf{0} \tag{21}$$

Adding and subtracting $\rho A^TA\boldsymbol{x}^{k+1}$ from the left hand side of (21), we obtain

$$\nabla h^{k+1}(\boldsymbol{x}^{k+1}) + A^T\boldsymbol{\lambda}^k + \rho A^TA\boldsymbol{x}^{k+1} + \rho(U + \bar{D} - A^TA)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) = \mathbf{0} \tag{22}$$

Recall that $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \tau\rho A\boldsymbol{x}^{k+1}$ and $Q = U + \bar{D} - A^TA$ hold, the above equality can be rewritten as

$$\nabla h^{k+1}(\boldsymbol{x}^{k+1}) + \rho(1-\tau)A^TA\boldsymbol{x}^{k+1} + A^T\boldsymbol{\lambda}^{k+1} + \rho Q(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) = \mathbf{0} \tag{23}$$

On the other hand, letting $(\boldsymbol{x}^*, \boldsymbol{\lambda}^{k+1*})$ be the Karush-Kuhn-Tucker (KKT) points for (5.13) at iteration $k+1$, we have

$$\begin{aligned} -A^T\boldsymbol{\lambda}^{k+1*} &= \nabla h^{k+1}(\boldsymbol{x}^*) \\ A\boldsymbol{x}^* &= \mathbf{0} \end{aligned} \tag{24}$$

which yields

$$\nabla h^{k+1}(\boldsymbol{x}^*) + A^T\boldsymbol{\lambda}^{k+1*} + \rho(1-\tau)A^TA\boldsymbol{x}^* = \mathbf{0} \tag{25}$$

It is worth noting that there may be more than one $\boldsymbol{\lambda}^{k+1*}$ satisfying (24). However, there is only one unique $\boldsymbol{\lambda}^{k+1*}$ lying in the column space of A [65]. In the following derivations, $\boldsymbol{\lambda}^{k+1*}$ indicates the one lying in the column space of A.

Subtracting (25) from (23), we obtain

$$\nabla h^{k+1}(\boldsymbol{x}^{k+1}) - \nabla h^{k+1}(\boldsymbol{x}^*) + \rho(1-\tau)A^TA(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) = -A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) - \rho Q(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \tag{26}$$

Based on Lemma 1, we have that

$$r^{k+1}(\boldsymbol{x}) = h^{k+1}(\boldsymbol{x}^{k+1}) + \frac{\rho(1-\tau)}{2} \parallel A\boldsymbol{x} \parallel^2$$

is restricted strongly convex with respect to $\boldsymbol{x}^*$, i.e.,

$$(\nabla h^{k+1}(\boldsymbol{x}^{k+1}) - \nabla h^{k+1}(\boldsymbol{x}^*) + \rho(1-\tau)A^TA(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*))^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) \geq m_r \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \tag{27}$$

Combing (26) and (27) leads to

$$-(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*)^TA^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) - \rho(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*)^TQ^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) \geq m_r \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \tag{28}$$

Moreover, we have the following equalities by using algebraic manipulations:

$$(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*)^TQ^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) = \frac{1}{2} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel_Q^2 + \frac{1}{2} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel_Q^2 - \frac{1}{2} \parallel \boldsymbol{x}^k - \boldsymbol{x}^* \parallel_Q^2, \tag{29}$$

$$\begin{aligned} (\boldsymbol{x}^{k+1} - \boldsymbol{x}^*)^TA^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) &= \frac{1}{\rho\tau}(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) \\ = \frac{1}{2\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \parallel^2 - \frac{1}{2\rho\tau} \parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k+1*} \parallel^2 &+ \frac{1}{2\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*} \parallel^2 \end{aligned} \tag{30}$$

Then using the above equalities, (28) can be rewritten as

$$\begin{aligned} m_r \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \leq & -\frac{\rho}{2} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel_Q^2 - \frac{1}{2\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*} \parallel^2 \\ & + \frac{\rho}{2} \parallel \boldsymbol{x}^k - \boldsymbol{x}^* \parallel_Q^2 + \frac{1}{2\rho\tau} \parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k+1*} \parallel^2 - \frac{\rho}{2} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel_Q^2 - \frac{1}{2\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \parallel^2 \end{aligned} \tag{31}$$

Recall that $H = \mathrm{diag}\{\rho Q, \frac{1}{\rho\tau}I_{D|E'|}\}$ and $\boldsymbol{y}^k = [\boldsymbol{x}^{kT}, \boldsymbol{\lambda}^{kT}]^T$, the above inequality can be simplified as

$$2m_r \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \leq \parallel \boldsymbol{y}^k - \boldsymbol{y}^{k+1*} \parallel_H^2 - \parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H^2 - \parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^k \parallel_H^2 \tag{32}$$

On the other hand, observe that for any constant $u > 1$, it holds that [65]

$$(u-1) \parallel \boldsymbol{a} - \boldsymbol{b} \parallel^2 \geq (1 - \frac{1}{u}) \parallel \boldsymbol{b} \parallel^2 - \parallel \boldsymbol{a} \parallel^2 \tag{33}$$

So we have

$$\begin{aligned} &(u-1) \parallel \nabla h^{k+1}(\boldsymbol{x}^{k+1}) - \nabla h^{k+1}(\boldsymbol{x}^*) \parallel^2 \\ &= (u-1) \parallel A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) + \rho Q^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) + \rho(1-\tau)A^TA(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) \parallel^2 \\ &\geq \frac{u-1}{u} \parallel A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) \parallel^2 - \parallel \rho Q^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) + \rho(1-\tau)A^TA(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) \parallel^2 \end{aligned} \tag{34}$$

Since $\boldsymbol{\lambda}^{k+1}$ and $\boldsymbol{\lambda}^{k+1*}$ lie in the column space of of A, we have [65]

$$\parallel A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*}) \parallel^2 \geq A_{\min} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*} \parallel^2 \tag{35}$$

and

$$\begin{aligned} &\parallel \rho Q^T(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k) + \rho(1-\tau)A^TA(\boldsymbol{x}^{k+1} - \boldsymbol{x}^*) \parallel^2 \\ &\leq 2\rho^2 Q_{\max} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel_Q^2 + 2\rho^2(1-\tau)^2 A_{\max}^2 \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \end{aligned} \tag{36}$$

where $Q_{\max}$ is the largest eigenvalue of Q, $A_{\min}$ is the smallest nonzero eigenvalue of A^TA, and $A_{\max}$ is the largest eigenvalue of A^TA.

In addition, we have

$$\parallel \nabla h^{k+1}(\boldsymbol{x}^{k+1}) - \nabla h^{k+1}(\boldsymbol{x}^*) \parallel^2 \leq L^2 \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2$$

according to Assumption 9. Therefore, based on (34)-(36), we can obtain

$$\begin{aligned} &(u-1)L^2 \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \\ &\geq \frac{(u-1)A_{\min}}{u} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*} \parallel^2 - 2\rho^2 Q_{\max} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel_Q^2 - 2\rho^2(1-\tau)^2 A_{\max}^2 \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 \end{aligned} \tag{37}$$

Using algebraic manipulations, the above inequality can be rewritten as

$$\begin{aligned} &(\frac{uL^2}{\rho\tau A_{\min}} + \frac{2\rho(1-\tau)^2 u A_{\max}^2}{(u-1)\tau A_{\min}}) \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^* \parallel^2 + \frac{2uQ_{\max}}{(u-1)\tau A_{\min}} \rho \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \parallel_Q^2 \\ &\geq \frac{1}{\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*} \parallel^2 \end{aligned} \tag{38}$$

Adding

$$\frac{2uQ_{\max}}{(u-1)\tau A_{\min}}\frac{1}{\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k} \parallel^2$$

and

$$\rho Q_{\max} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel^2$$

to the left hand side of the above inequality, and $\rho \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel_Q^2$ to the right hand side of the above inequality, we obtain the following inequality

$$\begin{aligned}&\frac{2uQ_{\max}}{(u-1)\tau A_{\min}}(\rho \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{k} \parallel_Q^2 + \frac{1}{\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k} \parallel^2)\\ &+ (\frac{uL^2}{\rho\tau A_{\min}} + \rho Q_{\max} + \frac{2\rho(1-\tau)^2 uA_{\max}^2}{(u-1)\tau A_{\min}}) \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel^2 \\ &\geq \frac{1}{\rho\tau} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^{k+1*} \parallel^2 + \rho \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel_Q^2\end{aligned} \tag{39}$$

based on the fact

$$\rho \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel_Q^2 \leq \rho Q_{\max} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel^2 .$$

Letting

$$\delta = \min\{\frac{(u-1)\tau A_{\min}}{2uQ_{\max}}, \frac{2m_r\rho\tau(u-1)A_{\min}}{\phi}\} \tag{40}$$

where

$$\phi = u(u-1)L^2 + \rho^2\tau A_{\min}Q_{\max}(u-1) + 2\rho^2(1-\tau)^2 uA_{\max}^2$$

we have from (39)

$$\frac{1}{\delta} \parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k} \parallel_H^2 + \frac{2m_r}{\delta} \parallel \boldsymbol{x}^{k+1} - \boldsymbol{x}^{*} \parallel^2 \geq \parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H^2 \tag{41}$$

Using (32) and (41), we can get

$$\frac{1}{\delta} \parallel \boldsymbol{y}^{k} - \boldsymbol{y}^{k+1*} \parallel_H^2 - \frac{1}{\delta} \parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H^2 \geq \parallel \boldsymbol{y}^{k+1} - \boldsymbol{y}^{k+1*} \parallel_H^2 \tag{42}$$

which proves Lemma 2.

B.2 Proof of Lemma 3

First, we have

$$\| \boldsymbol{y}^k - \boldsymbol{y}^{k+1*} \|_H - \| \boldsymbol{y}^k - \boldsymbol{y}^{k*} \|_H \leq \| \boldsymbol{y}^{k+1*} - \boldsymbol{y}^{k*} \|_H \tag{43}$$

On the other hand, we have

$$\| \boldsymbol{y}^{k+1*} - \boldsymbol{y}^{k*} \|_H = \frac{1}{\sqrt{\rho\tau}} \| \boldsymbol{\lambda}^{k+1*} - \boldsymbol{\lambda}^{k*} \| \tag{44}$$

$$\| A^T(\boldsymbol{\lambda}^{k+1*} - \boldsymbol{\lambda}^{k*}) \| = \| \triangledown h^{k+1}(\boldsymbol{x}^*) - \triangledown h^k(\boldsymbol{x}^*) \| \tag{45}$$

Therefore, using (35) one can obtain

$$\| \boldsymbol{\lambda}^{k+1*} - \boldsymbol{\lambda}^{k*} \| \leq \frac{1}{\sqrt{A_{\min}}} \| \triangledown h^{k+1}(\boldsymbol{x}^*) - \triangledown h^k(\boldsymbol{x}^*) \| \tag{46}$$

Combing (43) to (46) leads to

$$\| \boldsymbol{y}^k - \boldsymbol{y}^{k+1*} \|_H \leq \| \boldsymbol{y}^k - \boldsymbol{y}^{k*} \|_H + \frac{1}{\sqrt{\rho\tau A_{\min}}} \| \triangledown h^{k+1}(\boldsymbol{x}^*) - \triangledown h^k(\boldsymbol{x}^*) \| \tag{47}$$

which concludes the proof.

B.3 Proof of Theorem 12

From Lemma 4, we can obtain

$$\sqrt{1+\delta}^k \| \boldsymbol{y}^k - \boldsymbol{y}^{k*} \|_H \leq \| \boldsymbol{y}^0 - \boldsymbol{y}^{0*} \|_H + \sum_{s=0}^{k-1} \sqrt{1+\delta}^s p(s) \tag{48}$$

Dividing both sides of the above inequality by $\sqrt{1+\delta}^k$, we have

$$\| \boldsymbol{y}^k - \boldsymbol{y}^{k*} \|_H \leq \frac{\| \boldsymbol{y}^0 - \boldsymbol{y}^{0*} \|_H}{\sqrt{1+\delta}^k} + \sum_{s=0}^{k-1} \frac{1}{\sqrt{1+\delta}^{k-s}} p(s) \tag{49}$$

It is clear $\lim_{k\to\infty} \frac{\|\boldsymbol{y}^0-\boldsymbol{y}^{0*}\|_H}{\sqrt{1+\delta}^k} = 0$ as $\delta > 0$. Now our main goal reduces to proving $\lim_{k\to\infty} \sum_{s=0}^{k-1} \frac{1}{\sqrt{1+\delta}^{k-s}} p(s) = 0$. Recalling the relationship $\lim_{k\to\infty} f_i^{\alpha k} \to f_i^{\alpha *}$ from Assumption 9, we have $\lim_{k\to\infty} h_i^k \to h_i^*$ for all $i = 1, 2, \ldots, 2N$. On the other hand, according to the definition of $p(k)$ in (5.20), we know $\lim_{k\to\infty} p(k) = 0$ due to the convergence of h^k.

Therefore, we have that $p(k)$ is bounded, i,e., there exists a B such that $p(k) \leq B$ is true for an arbitrary k. In addition, we always have

$$\forall \varepsilon_1 > 0, \quad \exists N_1 \in \mathbb{N}^+, \quad \text{s.t.} \quad |p(k)| \leq \varepsilon_1, \quad \forall k \geq N_1,$$

where $\mathbb{N}^+$ is the set of positive integers. Further letting $\eta = \frac{1}{\sqrt{1+\delta}}$ and $F(k) = \sum_{s=0}^{k-1} \frac{1}{\sqrt{1+\delta}^{k-s}} p(s)$, we can obtain

$$\begin{aligned} F(k) &= \sum_{s=0}^{k-1} \eta^{k-s} p(s) \\ &= \sum_{s=0}^{N_1} \eta^{k-s} p(s) + \sum_{s=N_1+1}^{k-1} \eta^{k-s} p(s) \\ &\leq B \sum_{s=0}^{N_1} \eta^{k-s} + \varepsilon_1 \sum_{s=N_1+1}^{k-1} \eta^{k-s} \\ &= B\eta^k \frac{\eta^{-N_1} - \eta}{1-\eta} + \varepsilon_1 \frac{\eta - \eta^{k-N_1-1}}{1-\eta} \\ &\leq B\eta^k \frac{\eta^{-N_1} - \eta}{1-\eta} + \varepsilon_1 \frac{\eta}{1-\eta} \end{aligned} \tag{50}$$

for $k \geq N_1 + 2$ and $\eta \in (0, 1)$.

Therefore, we have $\lim_{k\to\infty} B\eta^k \frac{\eta^{-N_1}-\eta}{1-\eta} = 0$ and

$$\begin{aligned} &\forall \varepsilon = \varepsilon_1 > 0, \quad \exists N_2 \in \mathbb{N}^+, \\ &\text{s.t.} \quad |B\eta^k \frac{\eta^{-N_1} - \eta}{1-\eta}| \leq \varepsilon_1, \quad \forall k \geq N_2 \end{aligned} \tag{51}$$

Combining (50) and (51) leads to

$$\begin{aligned} &\forall \varepsilon = \varepsilon_1 > 0, \quad \exists N = \max\{N_1, N_2\}, \\ &\text{s.t.} \quad |F(k)| \leq \varepsilon_1 + \varepsilon_1 \frac{\eta}{1-\eta} = \frac{1}{1-\eta} \varepsilon_1, \quad \forall k \geq N \end{aligned} \tag{52}$$

which proves $\lim_{k\to\infty} F(k) = 0$ and further

$$\lim_{k\to\infty} \| \boldsymbol{y}^k - \boldsymbol{y}^{k*} \|_H = 0$$

based on (49). Given

$$\| \boldsymbol{x}^k - \boldsymbol{x}^* \|_Q \leq \| \boldsymbol{y}^k - \boldsymbol{y}^{k*} \|_H$$

we have $\lim_{k\to\infty} \| \boldsymbol{x}^k - \boldsymbol{x}^* \|_Q = 0$ as well, which concludes the proof.

Appendix C Proofs of Theorems in Chapter 7

C.1 Proof of (7.19) in Theorem 14

To prove (7.19) in Theorem 14, we first introduce two lemmas:

Lemma 13. *Let* $\boldsymbol{p}^k = [\boldsymbol{p}_1^{kT}, \boldsymbol{p}_2^{kT}, ..., \boldsymbol{p}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E}$ *be the iterates generated by GS-ADMM following* (7.15) *and* (7.16)*, then the following inequality holds for all* k*:*

$$f(\boldsymbol{p}) - f(\boldsymbol{p}^{k+1}) + (\boldsymbol{p} - \boldsymbol{p}^{k+1})^T J^T A^T \boldsymbol{\lambda}^{k+1} + \rho(\boldsymbol{p} - \boldsymbol{p}^{k+1})^T J^T (-A^T H + H^T H + I) J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \geq 0,$$
$$\forall \boldsymbol{p} \in \{[\boldsymbol{p}_1^T, \boldsymbol{p}_2^T, ..., \boldsymbol{p}_N^T]^T | \boldsymbol{p}_i \in \mathcal{P}_i, \forall i \in \{1, 2, ..., N\}\}, \tag{53}$$

where A *is the edge-node incident matrix defined in* (7.10)*,* $H = \min\{0, A\}$*, and* I *is the identity matrix. (In the following, we only consider* $\boldsymbol{p}$ *belonging to the set* $\{[\boldsymbol{p}_1^T, \boldsymbol{p}_2^T, ..., \boldsymbol{p}_N^T]^T | \boldsymbol{p}_i \in \mathcal{P}_i, \forall i \in \{1, 2, ..., N\}\}$*, so we leave out this constraint in the following lemmas and proofs.)*

Proof: Denote by g_i the function

$$\begin{aligned} g_i^k(\boldsymbol{p}_i) = &\sum_{j \in \hat{\mathcal{N}}_i, j \geq i} (\boldsymbol{\lambda}_{i,j}^{kT}(J_i \boldsymbol{p}_i - J_j \boldsymbol{p}_j^k) + \frac{\rho}{2} \parallel J_i \boldsymbol{p}_i - J_j \boldsymbol{p}_j^k \parallel^2) \\ &+ \sum_{j \in \hat{\mathcal{N}}_i, j < i} (\boldsymbol{\lambda}_{i,j}^{kT}(J_i \boldsymbol{p}_i - J_j \boldsymbol{p}_j^{k+1}) + \frac{\rho}{2} \parallel J_i \boldsymbol{p}_i - J_j \boldsymbol{p}_j^{k+1} \parallel^2). \end{aligned} \tag{54}$$

From the update rule in (7.15), we know that $\boldsymbol{p}_i^{k+1}$ is the optimizer of $g_i^k + f_i$ in the closed and convex set $\mathcal{P}_i$. Since f_i and g_i^k are convex, and g_i^k is differentiable, following the proof of Lemma 3.1 in [49] (which is also mentioned in Lemma 1 in [81]), we can get

$$f_i(\boldsymbol{p}_i) - f_i(\boldsymbol{p}_i^{k+1}) + (\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T \nabla g_i(\boldsymbol{p}_i^{k+1}) \geq 0. \tag{55}$$

Substituting $\nabla g_i(\boldsymbol{p}_i^{k+1})$ with (54), we have

$$\begin{aligned} f_i(\boldsymbol{p}_i) - f_i(\boldsymbol{p}_i^{k+1}) + (\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T (&\sum_{j \in \hat{\mathcal{N}}_i, j \geq i} (J_i^T \boldsymbol{\lambda}_{i,j}^k + \rho J_i^T (J_i \boldsymbol{p}_i^{k+1} - J_j \boldsymbol{p}_j^k))) \\ + (\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T (&\sum_{j \in \hat{\mathcal{N}}_i, j < i} (J_i^T \boldsymbol{\lambda}_{i,j}^k + \rho J_i^T (J_i \boldsymbol{p}_i^{k+1} - J_j \boldsymbol{p}_j^{k+1}))) \geq 0. \end{aligned}$$

Noting $\boldsymbol{\lambda}_{i,i} = 0$, using (7.16) leads to

$$f_i(\boldsymbol{p}_i) - f_i(\boldsymbol{p}_i^{k+1}) + (\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T(\sum_{j\in\mathcal{N}_i} J_i^T \boldsymbol{\lambda}_{i,j}^{k+1} + \sum_{j\in\hat{\mathcal{N}}_i, j\geq i} \rho J_i^T(J_j\boldsymbol{p}_j^{k+1} - J_j\boldsymbol{p}_j^k)) \geq 0.$$

Noting $\boldsymbol{\lambda}_{i,j} = -\boldsymbol{\lambda}_{j,i}$, from the definition of A, we can rewrite the above inequality as

$$f_i(\boldsymbol{p}_i) - f_i(\boldsymbol{p}_i^{k+1}) + (\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T(J_i^T[A]_i^T \boldsymbol{\lambda}^{k+1} + \sum_{j\in\hat{\mathcal{N}}_i, j\geq i} \rho J_i^T(J_j\boldsymbol{p}_j^{k+1} - J_j\boldsymbol{p}_j^k)) \geq 0, \tag{56}$$

here $[A]_i$ denotes the columns of A associated with cluster i.

Summing both sides of (56) over $i = 1, 2, ..., N$, and noticing that the following two equations hold [118],

$$\begin{aligned}
&\sum_{i=1}^{N}(\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T J_i^T[A]_i^T\boldsymbol{\lambda}^{k+1} = (J\boldsymbol{p} - J\boldsymbol{p}^{k+1})^T A^T\boldsymbol{\lambda}^{k+1}, \\
&\sum_{i=1}^{N}(\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T(\sum_{j\in\hat{\mathcal{N}}_i, j\geq i} \rho J_i^T(J_j\boldsymbol{p}_j^{k+1} - J_j\boldsymbol{p}_j^k)) \\
&= \rho(J\boldsymbol{p} - J\boldsymbol{p}^{k+1})^T[(-A + H)^T H + I](J\boldsymbol{p}^{k+1} - J\boldsymbol{p}^k),
\end{aligned} \tag{57}$$

we can get the lemma. ■

Lemma 14. *Let* $\boldsymbol{p}^k = [\boldsymbol{p}_1^{kT}, \boldsymbol{p}_2^{kT}, ..., \boldsymbol{p}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j}\in E}$ *be the iterates generated by GS-ADMM following* (7.15) *and* (7.16)*, then the following equality holds for all* k*:*

$$\begin{aligned}
&-(J\boldsymbol{p}^{k+1})^T A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \rho(J\boldsymbol{p}^* - J\boldsymbol{p}^{k+1})^T(H^T H - A^T H + I)J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \\
&= -\frac{1}{2\rho}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2 - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2) - \frac{\rho}{2}(\| HJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^*) \|^2 - \| HJ(\boldsymbol{p}^k - \boldsymbol{p}^*) \|^2) - \\
&\frac{\rho}{2}(\| J\boldsymbol{p}^{k+1} - J\boldsymbol{p}^* \|^2 - \| J\boldsymbol{p}^k - J\boldsymbol{p}^* \|^2) - \frac{\rho}{2} \| HJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) - AJ\boldsymbol{p}^{k+1} \|^2 - \frac{\rho}{2} \| J\boldsymbol{p}^{k+1} - J\boldsymbol{p}^k \|^2
\end{aligned} \tag{58}$$

Proof: Since for a scalar a, $a^T = a$ holds, and recall $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho AJ\boldsymbol{p}^{k+1}$, we can get

$$(\boldsymbol{p}^{k+1})^T J^T A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) = \frac{1}{\rho}(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k)^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*). \tag{59}$$

In addition, as $(\boldsymbol{p}^*, \boldsymbol{\lambda}^*)$ is the saddle point of the Lagrangian function $L(\boldsymbol{p}, \boldsymbol{\lambda}) = f(\boldsymbol{p}) + \boldsymbol{\lambda}^T AJ\boldsymbol{p}$, we have $AJ\boldsymbol{p}^*=0$. So we can establish the following relationships using algebraic manipulation:

$$(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k)^T(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^*)=\frac{1}{2}\parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k\parallel^2+\frac{1}{2}(\parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^*\parallel^2-\parallel \boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*\parallel^2), \tag{60}$$

$$(\boldsymbol{p}^{k+1}-\boldsymbol{p}^*)^TJ^TIJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)=\frac{1}{2}\parallel J\boldsymbol{p}^{k+1}-J\boldsymbol{p}^k\parallel^2+\frac{1}{2}(\parallel J\boldsymbol{p}^{k+1}-J\boldsymbol{p}^*\parallel^2-\parallel J\boldsymbol{p}^k-J\boldsymbol{p}^*\parallel^2), \tag{61}$$

$$\begin{aligned}&(\boldsymbol{p}^{k+1}-\boldsymbol{p}^*)^TJ^TH^THJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)\\&=\frac{1}{2}(\parallel HJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^*)\parallel^2-\parallel HJ(\boldsymbol{p}^k-\boldsymbol{p}^*)\parallel^2)+\frac{1}{2}\parallel HJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)\parallel^2,\end{aligned} \tag{62}$$

$$\begin{aligned}&(\boldsymbol{p}^{k+1}-\boldsymbol{p}^*)^TJ^TA^THJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)\\&=\frac{1}{2}\parallel HJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)\parallel^2+\frac{1}{2\rho^2}\parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^k\parallel^2-\frac{1}{2}\parallel HJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)-AJ\boldsymbol{p}^{k+1}\parallel^2.\end{aligned} \tag{63}$$

Then (58) can be proven by plugging equations (59) to (63) into the left part of (58). ■

Now we proceed to prove Theorem 14. Set $\boldsymbol{p}=\boldsymbol{p}^*$ in (53), and recall $AJ\boldsymbol{p}^*=0$, then we have

$$f(\boldsymbol{p}^*)-f(\boldsymbol{p}^{k+1})-\boldsymbol{p}^{(k+1)T}J^TA^T\boldsymbol{\lambda}^{k+1}+\rho(\boldsymbol{p}^*-\boldsymbol{p}^{k+1})^TJ^T(-A^TH+H^TH+I)J(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)\geq 0. \tag{64}$$

Adding and subtracting the term $\boldsymbol{\lambda}^{*T}AJ\boldsymbol{p}^{k+1}$ from the left side of (64), we can get

$$\begin{aligned}&f(\boldsymbol{p}^*)-f(\boldsymbol{p}^{k+1})-\boldsymbol{\lambda}^{*T}AJ\boldsymbol{p}^{k+1}-\boldsymbol{p}^{(k+1)T}J^TA^T(\boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^*)+\\&\rho(\boldsymbol{p}^*-\boldsymbol{p}^{k+1})^TJ^T(-A^TH+H^TH+I)J(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)\geq 0.\end{aligned}$$

Now by applying (58) into the above inequality, the following inequality can be obtained:

$$\begin{aligned}&f(\boldsymbol{p}^*)-f(\boldsymbol{p}^{k+1})-\boldsymbol{\lambda}^{*T}AJ\boldsymbol{p}^{k+1}-\frac{1}{2\rho}(\parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^*\parallel^2-\parallel \boldsymbol{\lambda}^k-\boldsymbol{\lambda}^*\parallel^2)\\&-\frac{\rho}{2}(\parallel HJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^*)\parallel^2-\parallel HJ(\boldsymbol{p}^k-\boldsymbol{p}^*)\parallel^2)-\frac{\rho}{2}(\parallel J\boldsymbol{p}^{k+1}-J\boldsymbol{p}^*\parallel^2-\parallel J\boldsymbol{p}^k-J\boldsymbol{p}^*\parallel^2)\\&-\frac{\rho}{2}\parallel HJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^k)-AJ\boldsymbol{p}^{k+1}\parallel^2-\frac{\rho}{2}\parallel J\boldsymbol{p}^{k+1}-J\boldsymbol{p}^k\parallel^2\geq 0.\end{aligned}$$

Summing both sides of the inequality over $k=0,1,...,t$, we can obtain the following result after some re-arrangement:

$$
\begin{aligned}
&(t+1)f(\boldsymbol{p}^*) - \sum_{k=0}^{t} f(\boldsymbol{p}^{k+1}) - \boldsymbol{\lambda}^{*T} AJ \sum_{k=0}^{t} \boldsymbol{p}^{k+1} \\
&+ \frac{\rho}{2}(\| HJ(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \| J\boldsymbol{p}^0 - J\boldsymbol{p}^* \|^2) + \frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 \\
&\geq \sum_{k=0}^{t} \frac{\rho}{2}(\| HJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) - AJ\boldsymbol{p}^{k+1} \|^2) + \sum_{k=0}^{t} \frac{\rho}{2}(\| J\boldsymbol{p}^{k+1} - J\boldsymbol{p}^k \|^2) + \frac{1}{2\rho} \| \boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^* \|^2 \\
&+ \frac{\rho}{2}(\| HJ(\boldsymbol{p}^{t+1} - \boldsymbol{p}^*) + \| J\boldsymbol{p}^{t+1} - J\boldsymbol{p}^* \|^2) \geq 0.
\end{aligned}
$$

In addition, as our function is convex, we have $\sum_{k=0}^{t} f(\boldsymbol{p}^{k+1}) \geq (t+1)f(\bar{\boldsymbol{p}}^{t+1})$, then we can get

$$
\begin{aligned}
&(t+1)f(\boldsymbol{p}^*) - (t+1)f(\bar{\boldsymbol{p}}^{t+1}) - (t+1)\boldsymbol{\lambda}^{*T} AJ\bar{\boldsymbol{p}}^{t+1} \\
&+ \frac{\rho}{2}(\| HJ(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \| J\boldsymbol{p}^0 - J\boldsymbol{p}^* \|^2) + \frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 \geq 0
\end{aligned}
$$

Dividing both sides by $-(t+1)$ yields

$$
\begin{aligned}
&f(\bar{\boldsymbol{p}}^{t+1}) + \boldsymbol{\lambda}^{*T} AJ\bar{\boldsymbol{p}}^{t+1} - f(\boldsymbol{p}^*) \\
&\leq \frac{\rho}{2(t+1)}(\| HJ(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \| J\boldsymbol{p}^0 - J\boldsymbol{p}^* \|^2) + \frac{1}{(t+1)2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 .
\end{aligned}
\tag{65}
$$

Combining the above relationship (65) with the Lagrangian function $L(\boldsymbol{p}, \boldsymbol{\lambda}) = f(\boldsymbol{p}) + \boldsymbol{\lambda}^T AJ\boldsymbol{p}$, (7.19) in Theorem 14 is proven. ■

C.2 Proof of Theorem 16

To prove Theorem 16, we first introduce two lemmas:

Lemma 15. *Let* $\boldsymbol{p}^k = [\boldsymbol{p}_1^{kT}, \boldsymbol{p}_2^{kT}, ..., \boldsymbol{p}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j} \in E}$ *be the iterates generated by J-ADMM following* (7.17) *and* (7.18)*, then the following inequality holds for all* k*:*

$$
f(\boldsymbol{p}) - f(\boldsymbol{p}^{k+1}) + (\boldsymbol{p} - \boldsymbol{p}^{k+1})^T J^T A^T \boldsymbol{\lambda}^{k+1} + \rho(\boldsymbol{p} - \boldsymbol{p}^{k+1})^T J^T(-A^T A + \bar{Q}^T \bar{Q})J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \geq 0,
\tag{66}
$$

where $\bar{Q}$ *is defined in (7.21).*

Proof: Denote by g_i the function

$$g_i^k(\boldsymbol{p}_i) = \sum_{j\in\hat{\mathcal{N}}_i} (\boldsymbol{\lambda}_{i,j}^{kT}(J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^k) + \frac{\rho}{2} \parallel J_i\boldsymbol{p}_i - J_j\boldsymbol{p}_j^k \parallel^2) + \frac{\rho\gamma_i}{2} \parallel J_i\boldsymbol{p}_i - J_i\boldsymbol{p}_i^k \parallel^2 . \tag{67}$$

Then following the proof of Lemma 13, we can get

$$f_i(\boldsymbol{p}_i) - f_i(\boldsymbol{p}_i^{k+1}) + (\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T J_i^T([A]_i^T\boldsymbol{\lambda}^{k+1} + \sum_{j\in\hat{\mathcal{N}}_i} \rho J_j(\boldsymbol{p}_j^{k+1} - \boldsymbol{p}_j^k) + \rho\gamma_i J_i(\boldsymbol{p}_i^{k+1} - \boldsymbol{p}_i^k)) \geq 0. \tag{68}$$

Summing both sides of the above relation over $i = 1, 2, ...N$, and noticing that the following two equations hold,

$$\begin{aligned}
&\sum_{i=1}^{N}(\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T J_i^T \rho(\sum_{j\in\hat{\mathcal{N}}_i} J_j(\boldsymbol{p}_j^{k+1} - \boldsymbol{p}_j^k) + \gamma_i J_i(\boldsymbol{p}_i^{k+1} - \boldsymbol{p}_i^k)) \\
&= \rho(\boldsymbol{p} - \boldsymbol{p}^{k+1})^T J^T[-A^TA + Q_C + I + Q_P]J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k), \\
&\sum_{i=1}^{N}(\boldsymbol{p}_i - \boldsymbol{p}_i^{k+1})^T J_i^T [A]_i^T \boldsymbol{\lambda}^{k+1} = (\boldsymbol{p} - \boldsymbol{p}^{k+1})^T J^T A^T \boldsymbol{\lambda}^{k+1},
\end{aligned}$$

we can get the lemma. ■

Lemma 16. *Let* $\boldsymbol{p}^k = [\boldsymbol{p}_1^{kT}, \boldsymbol{p}_2^{kT}, ..., \boldsymbol{p}_N^{kT}]^T$ *and* $\boldsymbol{\lambda}^k = [\boldsymbol{\lambda}_{i,j}^k]_{ij, e_{i,j}\in E}$ *be the iterates generated by J-ADMM following* (7.17) *and* (7.18). *Then the following equality holds for all k:*

$$\begin{aligned}
&-(\boldsymbol{p}^{k+1})^T J^T A^T(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^*) + \rho(\boldsymbol{p}^* - \boldsymbol{p}^{k+1})^T J^T(-A^TA + \bar{Q}^T\bar{Q})J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \\
&= -\frac{1}{2\rho}(\parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \parallel^2 - \parallel \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \parallel^2) + \frac{\rho}{2}(\parallel AJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^*) \parallel^2 - \parallel AJ(\boldsymbol{p}^k - \boldsymbol{p}^*) \parallel^2) \\
&- \frac{\rho}{2}(\parallel \bar{Q}J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^*) \parallel^2 - \parallel \bar{Q}J(\boldsymbol{p}^k - \boldsymbol{p}^*) \parallel^2) + \frac{\rho}{2} \parallel AJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \parallel^2 - \frac{\rho}{2} \parallel \bar{Q}J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \parallel^2 \\
&- \frac{1}{2\rho} \parallel \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \parallel^2 .
\end{aligned} \tag{69}$$

Proof: The proof is similar to the proof of Lemma 14 and is omitted. ■

Then following the proof of Theorem 14 (setting $\boldsymbol{p} = \boldsymbol{p}^*$ in (66) and applying (69)), we can obtain the following inequality:

$$
\begin{aligned}
& f(\boldsymbol{p}^*) - f(\boldsymbol{p}^{k+1}) - \boldsymbol{\lambda}^{*T} A J \boldsymbol{p}^{k+1} - \frac{1}{2\rho}(\| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^* \|^2 - \| \boldsymbol{\lambda}^k - \boldsymbol{\lambda}^* \|^2) \\
& + \frac{\rho}{2}(\| AJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^*) \|^2 - \| AJ(\boldsymbol{p}^k - \boldsymbol{p}^*) \|^2) - \frac{\rho}{2}(\| \bar{Q}J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^*) \|^2 - \| \bar{Q}J(\boldsymbol{p}^k - \boldsymbol{p}^*) \|^2) \\
& + \frac{\rho}{2} \| AJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \|^2 - \frac{\rho}{2} \| \bar{Q}J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \|^2 - \frac{1}{2\rho} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2 \geq 0. \qquad (70)
\end{aligned}
$$

Summing both sides of the above inequality over $k = 0, 1, ..., t$, we can get the following result after some re-arrangement:

$$
\begin{aligned}
& (t+1)f(\boldsymbol{p}^*) - \sum_{k=0}^{t} f(\boldsymbol{p}^{k+1}) - \boldsymbol{\lambda}^{*T} AJ \sum_{k=0}^{t} \boldsymbol{p}^{k+1} + \frac{\rho}{2} \| \bar{Q}J(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 \\
& \geq \frac{\rho}{2} \| AJ(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \frac{1}{2\rho} \| \boldsymbol{\lambda}^{t+1} - \boldsymbol{\lambda}^* \|^2 + \sum_{k=0}^{t} \frac{\rho}{2}(\| \bar{Q}J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \|^2 - \| AJ(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \|^2) \\
& + \frac{\rho}{2}(\| \bar{Q}J(\boldsymbol{p}^{t+1} - \boldsymbol{p}^*) \|^2 - \| AJ(\boldsymbol{p}^{t+1} - \boldsymbol{p}^*) \|^2) + \sum_{k=0}^{t} \frac{1}{2\rho} \| \boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k \|^2 \\
& \geq \sum_{k=0}^{t} \frac{\rho}{2}(\| \bar{Q}J(\boldsymbol{p}^{k+1} - \boldsymbol{p}^k) \|^2 - \| A \|^2 \| J\boldsymbol{p}^{k+1} - J\boldsymbol{p}^k \|^2) \\
& + \frac{\rho}{2}(\| \bar{Q}J(\boldsymbol{p}^{t+1} - \boldsymbol{p}^*) \|^2 - \| A \|^2 \| J\boldsymbol{p}^{t+1} - J\boldsymbol{p}^* \|^2).
\end{aligned}
$$

Since $\| A \|^2 = \alpha_{\max}$, $\bar{Q}$ is a diagonal matrix with $\gamma_i' \geq \sqrt{\alpha_{\max}}$, we can get that the right hand side of the above inequality is greater than 0, which leads to

$$
(t+1)f(\boldsymbol{p}^*) - \sum_{k=0}^{t} f(\boldsymbol{p}^{k+1}) - \boldsymbol{\lambda}^{*T} AJ \sum_{k=0}^{t} \boldsymbol{p}^{k+1} + \frac{\rho}{2} \| \bar{Q}J(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 \geq 0.
$$

In addition, as our function is convex, we have $\sum_{k=0}^{t} f(\boldsymbol{p}^{k+1}) \geq (t+1) f(\bar{\boldsymbol{p}}^{t+1})$ and

$$
(t+1)f(\boldsymbol{p}^*) - (t+1)f(\bar{\boldsymbol{p}}^{t+1}) - (t+1)\boldsymbol{\lambda}^{*T} AJ\bar{\boldsymbol{p}}^{t+1} + \frac{\rho}{2} \| \bar{Q}J(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2 + \frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 \geq 0.
$$

By dividing both sides by $-(t+1)$, we can obtain

$$
f(\bar{\boldsymbol{p}}^{t+1}) + \boldsymbol{\lambda}^{*T} AJ\bar{\boldsymbol{p}}^{t+1} - f(\boldsymbol{p}^*) \leq \frac{1}{t+1}(\frac{1}{2\rho} \| \boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^* \|^2 + \frac{\rho}{2} \| \bar{Q}J(\boldsymbol{p}^0 - \boldsymbol{p}^*) \|^2).
$$

Combining the above relationship with the Lagrangian function $L(\boldsymbol{p}, \boldsymbol{\lambda}) = f(\boldsymbol{p}) + \boldsymbol{\lambda}^T AJ\boldsymbol{p}$, we can get statement (1) of Theorem 16.

In addition, from (70), we have

$$\frac{\rho}{2} \parallel AJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^{k}) \parallel^2 - \frac{\rho}{2} \parallel \bar{Q}J(\boldsymbol{p}^{k+1}-\boldsymbol{p}^{k}) \parallel^2 - \frac{1}{2\rho}(\parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{*} \parallel^2 - \parallel \boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{*} \parallel^2)$$
$$+\frac{\rho}{2}(\parallel AJ(\boldsymbol{p}^{k+1}-\boldsymbol{p}^{*}) \parallel^2 - \parallel AJ(\boldsymbol{p}^{k}-\boldsymbol{p}^{*}) \parallel^2) - \frac{\rho}{2}(\parallel \bar{Q}J(\boldsymbol{p}^{k+1}-\boldsymbol{p}^{*}) \parallel^2 - \parallel \bar{Q}J(\boldsymbol{p}^{k}-\boldsymbol{p}^{*}) \parallel^2)$$
$$-\frac{1}{2\rho} \parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{k} \parallel^2 \geq -f(\boldsymbol{p}^{*}) + f(\boldsymbol{p}^{k+1}) + \boldsymbol{\lambda}^{*T}AJ\boldsymbol{p}^{k+1} \geq 0$$

Rewrite the above inequality as:

$$\frac{1}{2\rho} \parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{*} \parallel^2 + \frac{\rho}{2} \parallel J(\boldsymbol{p}^{k+1}-\boldsymbol{p}^{*}) \parallel^2_{\bar{Q}^T\bar{Q}-A^TA}$$
$$\leq \frac{1}{2\rho} \parallel \boldsymbol{\lambda}^{k}-\boldsymbol{\lambda}^{*} \parallel^2 + \frac{\rho}{2} \parallel J(\boldsymbol{p}^{k}-\boldsymbol{p}^{*}) \parallel^2_{\bar{Q}^T\bar{Q}-A^TA} - \frac{1}{2\rho} \parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{k} \parallel^2 - \frac{\rho}{2} \parallel J(\boldsymbol{p}^{k}-\boldsymbol{p}^{k}) \parallel^2_{\bar{Q}^T\bar{Q}-A^TA} \tag{71}$$

Then we have

$$\lim_{k\to\infty}(\frac{1}{2\rho} \parallel \boldsymbol{\lambda}^{k+1}-\boldsymbol{\lambda}^{k} \parallel^2 + \frac{\rho}{2} \parallel J(\boldsymbol{p}^{k}-\boldsymbol{p}^{k}) \parallel^2_{\bar{Q}^T\bar{Q}-A^TA}) = 0 \tag{72}$$

Following a similar proof of Theorem 1, we can obtain statement (2) of Theorem 16. ■

Bibliography

[1] *URL: https://github.com/zhangchunlei0813/Nonconvex-localization*.

[2] Python-paillier lib. *URL: http://python-paillier.readthedocs.io/en/latest/phe.html*.

[3] *Paillier Library*, http://acsc.cs.utexas.edu/libpaillier/.

[4] A. Al-Anwar, Y. Shoukry, S. Chakraborty, P. Martin, P. Tabuada, and M. B. Srivastava. PrOLoc: resilient localization with private observers using partial homomorphic encryption. In *IPSN*, pages 41–52, 2017.

[5] A. Alaeddini, K. Morgansen, and M. Mesbahi. Adaptive communication networks with privacy guarantees. In *American Control Conference*, pages 4460–4465, 2017.

[6] A. Alanwar, Y. Shoukry, S. Chakraborty, B. Balaji, P. Martin, P. Tabuada, and M. Srivastava. PrOLoc: resilient localization with private observers using partial homomorphic encryption: demo abstract. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 257–258, 2017.

[7] R. Andreani, E. Birgin, J. M. Martínez, and M. L. Schuverdt. On augmented lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization*, 18(4):1286–1309, 2007.

[8] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt. Augmented lagrangian methods under the constant positive linear dependence constraint qualification. *Mathematical Programming*, 111(1-2):5–32, 2008.

[9] J. Bachrach and C. Taylor. Localization in sensor networks. *Handbook of sensor networks: Algorithms and Architectures*, 1, 2005.

[10] A. Beck, P. Stoica, and J. Li. Exact and approximate solutions of source localization problems. *IEEE Transactions on Signal Processing*, 56(5):1770–1778, 2008.

[11] A. Beck, M. Teboulle, and Z. Chikishev. Iterative minimization schemes for solving the single source localization problem. *SIAM Journal on Optimization*, 19(3):1397–1416, 2008.

[12] J. Bethencourt. Advanced crypto software collection. *URL: http://acsc. cs. utexas. edu/libpaillier*.

[13] D. Blatt and A. Hero. Energy-based sensor network source localization via projection onto convex sets. *IEEE Transactions on Signal Processing*, 54(9):3614–3619, 2006.

[14] J. A. Bondy and U. S. R. Murty. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[15] P. Bonnet, J. Gehrke, and P. Seshadri. Querying the physical world. *IEEE Personal Communications*, 7(5):10–15, 2000.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[17] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. *Lecture notes of EE3920, Stanford University*, 2003.

[18] G. Cakiades, S. Desai, S. Deligeorges, B. E. Buckland, and J. George. Fusion solution for soldier wearable gunfire detection systems. In *SPIE Defense, Security, and Sensing*, pages 838802–838802, 2012.

[19] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Transactions on parallel and distributed systems*, 25(1):222–233, 2014.

[20] D. Catalano, R. Gennaro, and N. Howgrave-Graham. The bit security of pailliers encryption scheme and its applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 229–243, 2001.

[21] V. Cevher, M. F. Duarte, and R. G. Baraniuk. Distributed target localization via spatial sparsity. In *Proceedings of the 16th European Signal Processing Conference*, pages 1–5, 2008.

[22] C. Chen, B. S. He, Y. Ye, and X. M. Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016.

[23] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[24] M. H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

[25] S. Deligeorges, G. Cakiades, J. George, Y. Q. Wang, and F. J. Doyle. A mobile self synchronizing smart sensor array for detection and localization of impulsive threat sources. In *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 351–356, 2015.

[26] W. Deng, M. Lai, Z. Peng, and W. Yin. Parallel multi-block ADMM with O (1/k) convergence. *Journal of Scientific Computing*, 71(2):712–736, 2017.

[27] W. Du, Y. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 222–233, 2004.

[28] T. Erseghe. A distributed and maximum-likelihood sensor network localization algorithm based upon a nonconvex problem formulation. *IEEE Transactions on Signal and Information Processing over Networks*, 1(4):247–258, 2015.

[29] N. M. Freris and P. Patrinos. Distributed computing over encrypted data. In *54th Annual Allerton Conference on Communication, Control, and Computing*, pages 1116–1122. IEEE, 2016.

[30] X. Fu, F. Chan, W. K. Ma, and H. C. So. A complex-valued semidefinite relaxation approach for two-dimensional source localization using distance measurements and imperfect receiver positions. In *2012 IEEE 11th International Conference on Signal Processing*, volume 2, pages 1491–1494, 2012.

[31] S. Gade and N. H. Vaidya. Private learning on networks: Part II. *arXiv preprint arXiv:1703.09185*, 2017.

[32] M. Gavish and A. J. Weiss. Performance analysis of bearing-only target location algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 28(3):817–828, 1992.

[33] C. Gentry et al. Fully homomorphic encryption using ideal lattices. In *STOC*, volume 9, pages 169–178, 2009.

[34] J. George and L. M. Kaplan. Shooter localization using a wireless sensor network of soldier-worn gunfire detection systems. *Journal of Advances in Information Fusion*, 8(1):15–32, 2013.

[35] 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*, 30(5):107–128, 2013.

[36] O. Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[37] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. Johns Hopkins University Press, 2012.

[38] M. T. Hale and M. Egerstedt. Differentially private cloud-based multi-agent optimization with constraints. In *American Control Conference*, pages 1235–1240, 2015.

[39] M. T. Hale and M. Egerstedt. Cloud-enabled differentially private multi-agent optimization with constraints. *IEEE Transactions on Control of Network Systems*, 2017.

[40] D. Han and X. Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.

[41] S. Han, W. K. Ng, L. Wan, and V. Lee. Privacy-preserving gradient-descent methods. *IEEE Transactions on Knowledge and Data Engineering*, 22(6):884–899, 2010.

[42] S. Han, U. Topcu, and G. J. Pappas. Differentially private distributed constrained optimization. *IEEE Transactions on Automatic Control*, 62(1):50–64, 2017.

[43] R. Hasan, R. Bobba, and H. Khurana. Analyzing NASPInet data flows. In *Proceedings of IEEE Power Systems Conference and Exposition*, pages 1–6, 2009.

[44] B. S. He, L. S. Hou, and X. M. Yuan. On full jacobian decomposition of the augmented lagrangian method for separable convex programming. *SIAM Journal on Optimization*, 25(4):2274–2312, 2015.

[45] B. S. He, L. Z. Liao, D. Han, and H. Yang. A new inexact alternating directions method for monotone variational inequalities. *Mathematical Programming*, 92(1):103–118, 2002.

[46] B. S. He, M. Tao, and X. M. Yuan. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, 22(2):313–340, 2012.

[47] B. S. He, H. K. Xu, and X. M. Yuan. On the proximal jacobian decomposition of ALM for multiple-block separable convex minimization problems and its relationship to ADMM. *Journal of Scientific Computing*, 66(3):1204–1217, 2016.

[48] B. S. He and H. Yang. Some convergence properties of a method of multipliers for linearly constrained monotone variational inequalities. *Operations research letters*, 23(3):151–161, 1998.

[49] B. S. He and X. M. Yuan. On the o(1/n) convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[50] J. Heidemann, Y. Li, A. Syed, J. Wills, and W. Ye. Underwater sensor networking: Research challenges and potential applications. *Technical Report ISI-TR-2005-603, USC/Information Sciences Institute*, 2005.

[51] K. C. Ho and W. Xu. An accurate algebraic solution for moving source location using TDOA and FDOA measurements. *IEEE Transactions on Signal Processing*, 52(9):2453–2463, 2004.

[52] M. Hong and Z. Q. Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017.

[53] M. Y. Hong, Z. Q. Luo, and M. Razaviyay. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3836–3840, 2015.

[54] Z. Huang, S. Mitra, and N. Vaidya. Differentially private distributed optimization. In *Proceedings of the 2015 International Conference on Distributed Computing and Networking*, page 4, 2015.

[55] S. Jain, S. J. Kim, and G. B. Giannakis. Backhaul-constrained multicell cooperation leveraging sparsity and spectral clustering. *IEEE Transactions on Wireless Communications*, 15(2):899–912, 2016.

[56] T. Jia and R. M. Buehrer. A set-theoretic approach to collaborative position location for wireless networks. *IEEE Transactions on Mobile Computing*, 10(9):1264–1275, 2011.

[57] L. M. Kaplan, Q. Le, and P. Molnár. Maximum likelihood methods for bearings-only target localization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 3001–3004, 2001.

[58] S. Kontogiorgis and R. Meyer. A variable-penalty alternating directions method for convex optimization. *Mathematical Programming*, 83(1-3):29–53, 1998.

[59] S. Kumar, R. Jain, and K. Rajawat. Asynchronous optimization over heterogeneous networks via consensus admm. *IEEE Transactions on Signal and Information Processing over Networks*, 2016.

[60] D. Labit and K. Taitz. Users guide for sedumi interface 1.04. 2002.

[61] R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation. *IEEE Signal Processing Magazine*, 30(1):82–105, 2013.

[62] F. Li, B. Luo, and P. Liu. Secure information aggregation for smart grids using homomorphic encryption. In *2010 First IEEE International Conference on Smart Grid Communications*, pages 327–332, 2010.

[63] J. Lin, A. S. Morse, and B. Anderson. The multi-agent rendezvous problem-the asynchronous case. In *43rd IEEE Conference on Decision and Control*, volume 2, pages 1926–1931, 2004.

[64] Q. Ling, Y. Liu, W. Shi, and Z. Tian. Weighted admm for fast decentralized network optimization. *IEEE Transactions on Signal Processing*, 64(22):5930–5942, 2016.

[65] Q. Ling and A. Ribeiro. Decentralized dynamic optimization through the alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 62(5):1185–1197, 2014.

[66] Q. Ling, W. Shi, G. Wu, and A. Ribeiro. Dlm: Decentralized linearized alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(15):4051–4064, 2015.

[67] J. Liu, J. Chen, and J. Ye. Large-scale sparse logistic regression. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–556, 2009.

[68] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on knowledge and Data Engineering*, 18(1):92–106, 2006.

[69] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *IEEE Transactions on Automatic Control*, 56(6):1291–1306, 2011.

[70] J. Löfberg. Yalmip: A toolbox for modeling and optimization in MATLAB. In *Proceedings of IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, 2004.

[71] Y. Lou, L. Yu, S. Wang, and P. Yi. Privacy preservation in distributed subgradient optimization algorithms. *IEEE Transactions on Cybernetics*, 2017.

[72] S. Magnússon, P. Chathuranga, M. Rabbat, and C. Fischione. On the convergence of alternating direction lagrangian methods for nonconvex structured optimization problems. 2014.

[73] S. Magnússon, P. C. Weeraddana, and C. Fischione. A distributed approach for the optimal power-flow problem based on ADMM and sequential convex approximations. *IEEE Transactions on Control of Network Systems*, 2(3):238–253, 2015.

[74] O. L. Mangasarian. Privacy-preserving horizontally partitioned linear programs. *Optimization Letters*, 6(3):431–436, 2012.

[75] N. E. Manitara and C. N. Hadjicostis. Privacy-preserving asymptotic average consensus. In *2013 European Control Conference*, pages 760–765, 2013.

[76] M. Maros and J. Jaldén. On the Q-linear convergence of distributed generalized ADMM under non-strongly convex function components. *IEEE Transactions on Signal and Information Processing over Networks*, 2019.

[77] G. Mateos, J. A. Bazerque, and G. B. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276, 2010.

[78] W. Meng, W. Xiao, L. Xie, et al. A projection based fully distributed approach for source localization in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 18(1-2):131–158, 2013.

[79] W. Meng, W. Xiao, L. Xie, and A. Pandharipande. Diffusion based projection method for distributed source localization in wireless sensor networks. In *2011 IEEE Conference on Computer Communications Workshops*, pages 537–542, 2011.

[80] Y. Mo and R. M. Murray. Privacy preserving average consensus. *IEEE Transactions on Automatic Control*, 62(2):753–765, 2017.

[81] J. Mota, J. Xavier, P. Aguiar, and M. Püschel. A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. *arXiv preprint arXiv:1112.2295*, 2011.

[82] J. Mota, J. Xavier, P. Aguiar, and M. Puschel. D-ADMM: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing*, 61(10):2718–2723, 2013.

[83] S. Nabavi, J. H. Zhang, and A. Chakrabortty. Distributed optimization algorithms for wide-area oscillation monitoring in power systems using interregional PMU-PDC architectures. *IEEE Transactions on Smart Grid*, 6(5):2529–2538, 2015.

[84] A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

[85] A. Nedic, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

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

[87] A. Nedic, A. Ozdaglar, and P. A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[88] A. Nedic, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[89] E. Nozari, P. Tallapragada, and J. Cortes. Differentially private distributed convex optimization via functional perturbation. *IEEE Transactions on Control of Network Systems*, 2016.

[90] F. Núñez, Y. Q. Wang, D. Grasing, S. Desai, G. Cakiadesc, and F. J. Doyle. Pulse-coupled time synchronization for distributed acoustic event detection using wireless sensor networks. *Control Engineering Practice*, 60:106–117, 2017.

[91] P. Oğuz-Ekim, J. Gomes, J. Xavier, P. Oliveira, et al. A convex relaxation for approximate maximum-likelihood 2D source localization from range measurements. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2698–2701, 2010.

[92] P. Oğuz-Ekim, J. Gomes, J. Xavier, M. Stošić, and P. Oliveira. An angular approach for range-based approximate maximum likelihood source localization through convex relaxation. *IEEE Transactions on Wireless Communications*, 13(7):3951–3964, 2014.

[93] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[94] D. V. Ouellette. Schur complements and statistics. *Linear Algebra and its Applications*, 36:187–295, 1981.

[95] A. Papadogiannis, D. Gesbert, and E. Hardouin. A dynamic clustering approach in wireless networks with multi-cell cooperative processing. In *2008 IEEE International Conference on Communications*, pages 4033–4037, 2008.

[96] S. Pequito, S. Kar, S. Sundaram, and A. P. Aguiar. Design of communication networks for distributed computation with privacy guarantees. In *2014 IEEE 53rd Annual Conference on Decision and Control*, pages 1370–1376, 2014.

[97] N. Piovesan and T. Erseghe. Cooperative localization in WSNs: a hybrid convex/non-convex solution. *IEEE Transactions on Signal and Information Processing over Networks*, 2016.

[98] M. H. Ruan, M. Ahmad, and Y. Q. Wang. Secure and privacy-preserving average consensus. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*, pages 123–129, 2017.

[99] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[100] R. O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[101] Q. Shi and C. He. Distributed source localization via projection onto the nearest local minimum. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2553–2556, 2008.

[102] W. Shi, Q. Ling, G. Wu, and W. Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[103] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7):1750–1761, 2014.

[104] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada. Privacy-aware quadratic optimization using partially homomorphic encryption. In *2016 IEEE 55th Conference on Decision and Control*, pages 5053–5058. IEEE, 2016.

[105] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12, 2004.

[106] A. Simonetto and G. Leus. Distributed maximum likelihood sensor network localization. *IEEE Transactions on Signal Processing*, 62(6):1424–1437, 2014.

[107] A. X. Sun, D. T. Phan, and S. Ghosh. Fully decentralized AC optimal power flow algorithms. In *Power and Energy Society General Meeting (PES)*, pages 1–5, 2013.

[108] M. Tao and X. M. Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011.

[109] S. Venkatesan. Coordinating base stations for greater uplink spectral efficiency in a cellular network. In *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, 2007.

[110] H. Waki. How to generate weakly infeasible semidefinite programs via lasserres relaxations for polynomial optimization. *Optimization Letters*, pages 1–14.

[111] C. Wang, K. Ren, and J. Wang. Secure and practical outsourcing of linear programming in cloud computing. In *2011 Proceedings IEEE INFOCOM*, pages 820–828. IEEE, 2011.

[112] C. L. Wang, D. S. Wu, S. C. Chen, and K. J. Yang. A decentralized positioning scheme based on recursive weighted least squares optimization for wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 64(10):4887–4893, 2015.

[113] G. Wang. A semidefinite relaxation method for energy-based source localization in sensor networks. *IEEE Transactions on Vehicular Technology*, 60(5):2293–2301, 2011.

[114] G. Wang, Y. Li, and R. Wang. New semidefinite relaxation method for acoustic energy-based source localization. *IEEE Sensors Journal*, 13(5):1514–1521, 2013.

[115] J. Wang and P. Regalia. Sensor network localization via boundary projections. In *CISS*, pages 224–229, 2009.

[116] Y. Wang, W. Yin, and J. Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*, 2015.

[117] Y. Q. Wang and J. P. Hespanha. Distributed estimation of power system oscillation modes under attacks on gps clocks. *IEEE Transactions on Instrumentation and Measurement*, 2018.

[118] E. Wei and A. Ozdaglar. Distributed alternating direction method of multipliers. In *Proceedings of the 51st IEEE Conference on Decision and Control*, pages 5445–5450, 2012.

[119] E. Wei and A. Ozdaglar. On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 551–554, 2013.

[120] Z. Wen, C. Yang, X. Liu, and S. Marchesini. Alternating direction methods for classical and ptychographic phase retrieval. *Inverse Problems*, 28(11):115010, 2012.

[121] X. Xu, S. Sahni, and N. Rao. On basic properties of localization using distance-difference measurements. In *2008 11th International Conference on Information Fusion*, pages 1–8, 2008.

[122] Z. Xu and Q. Zhu. Secure and resilient control design for cloud enabled networked control systems. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, pages 31–42, 2015.

[123] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi. Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2483–2493, 2013.

[124] K. Yang, G. Wang, and Z. Q. Luo. Efficient convex relaxation methods for robust target localization by a sensor network using time differences of arrivals. *IEEE Transactions on Signal Processing*, 57(7):2775–2784, 2009.

[125] A. C. Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[126] J. Yuan, W. Ai, H. Deng, T. Shuai, and X. Zhao. Exact solution of an approximate weighted least squares estimate of energy-based source localization in sensor networks. *IEEE Transactions on Vehicular Technology*, 64(10):4645–4654, 2015.

[127] K. Yuan, Q. Ling, and Z. Tian. Communication-efficient decentralized event monitoring in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(8):2198–2207, 2015.

[128] F. Zeng, C. Li, and Z. Tian. Distributed compressive spectrum sensing in cooperative multihop cognitive networks. *IEEE Journal of Selected Topics in Signal Processing*, 5(1):37–48, 2011.

[129] C. L. Zhang, M. Ahmad, and Y. Q. Wang. Admm based privacy-preserving decentralized optimization. *IEEE Transactions on Information Forensics and Security*, 14(3):565–580, 2018.

[130] C. L. Zhang and Y. Q. Wang. Distributed event localization via alternating direction method of multipliers. *IEEE Transactions on Mobile Computing*, 17(2):348–361, 2017.

[131] C. L. Zhang and Y. Q. Wang. Enabling privacy-preservation in decentralized optimization. *IEEE Transactions on Control of Network Systems*, 2018.

[132] C. L. Zhang and Y. Q. Wang. Sensor network event localization via non-convex non-smooth admm and augmented lagrangian methods. *IEEE Transactions on Control of Network Systems*, 2019.

[133] J. Zhang, R. Chen, J. G. Andrews, A. Ghosh, and R. W. Heath. Networked MIMO with clustered linear precoding. *IEEE Transactions on Wireless Communications*, 8(4):1910–1921, 2009.

[134] T. Zhang and Q. Zhu. Dynamic differential privacy for ADMM-based distributed classification learning. *IEEE Transactions on Information Forensics and Security*, 12(1):172–187, 2017.

[135] Y. Zhang, Y. Lou, Y. Hong, and L. Xie. Distributed projection-based algorithms for source localization in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 14(6):3131–3142, 2015.

[136] S. Zhu, M. Hong, and B. Chen. Quantized consensus ADMM for multi-agent distributed optimization. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4134–4138, 2016.