Modeling and Simulation of Multi-Task Networks Using Adaptation and Learning

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von
Ing. Sahar Khawatmi
geboren am 15. März 1986 in Bielefeld (Deutschland)

Referent:	Prof. Dr.-Ing. Abdelhak M. Zoubir
Korreferent:	Prof. Ali H. Sayed
Tag der Einreichung:	08. Mai 2017
Tag der mündlichen Prüfung:	20. September 2017

D 17
Darmstädter Dissertation 2017

'A boat does not sail on land.'

Muhammad ibn Idris al-Shafi'i

To those who believe in me.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr.-Ing. Abdelhak M. Zoubir for his patience, motivation, immense knowledge, and the continuous support of my PhD study and related research. His guidance helped me throughout my research time and for the completeness of my PhD thesis. I would also like to thank my co-supervisor, Prof. Ali H. Sayed, for the insightful discussion, offering valuable advice, and especially for his patience and guidance during the writing process. I would also like to extend my thanks to my doctoral examiners, Prof. Dr. Mario Kupnik, Prof. Dr. rer. nat. Andy Schürr, and Prof. Dr.-Ing. Marius Pesavento.

My appreciation likewise extends to all current and former members of the Signal Processing Group. Words cannot express how grateful I am to Lala Khadidja Hamaidi and Freweyni Kidane Teklehaymanot for the friendship, spiritual support, and all the fun we have had in our office, to Sara Al-Sayed for assisting me in many different ways, to Renate Koschella for her innumerable help and support, to Mouhammad Alhumaidi, Patricia Binder, Nevine Demitri, Hauke Fath, Raquel Fandos, Michael Fauss, Gökhan Gül, Jürgen Hahn, Philipp Heidenreich, Roy Howard, Di Jin, Michael Lang, Michael Leigsnering, Stefan Leier, Mark Ryan Leonard, Zhihua Lu, Michael Muma, Ahmed Moustafa, Ivana Perna, Dominik Reinhard, Simon Rosenkranz, Tim Schäck, Ann-Kathrin Seifert, Waqas Sharif, Sergey Sukhanov, Adrian Šošić, Wassim Suleiman, Fiky Suratman, Gebremichael Teame, Christian Weiss, and Feng Yin for providing a nice atmosphere in our group.

I wish to thank my parents, Ahed and Maher, my brother Sinan, and my sister Sana for their endless and unconditional support, love, and encouragement. I am also greatful to my parents-in-law, Ieman and Faissal, my aunts Essam, Manhal, and Manal, and my extended family, Ola, Ranim, Musab, Noor, Adnan, Rahaf, Yaman, and Hala for their moral supporting and permanent wishes.

I want to express my gratitude and deepest appreciation to Alia, Aya, Hala, Heba, and Ruba for being beside me through thick and thin, I find myself lucky to have friends like them in my life. Special thanks go to Heba, without our daily discussions during the preparation of this thesis, I would not have been progressed.

Last, but never least, I would like to thank my loving, encouraging, and patient husband Hosam for his faithful support over these years. And to my pretty daughter Mariam, thank you for everything that you are, and everything you will become.

Darmstadt, 22.09.2017

Kurzfassung

Die vorliegende Doktorarbeit beschäftigt sich mit kooperativen Netzwerken mit mehreren Aufgaben. Kooperative Netzwerke bestehen aus einer Sammlung von Agenten mit Adaptions- und Lernfähigkeiten. Die Idee, Daten zwischen benachbarten Agenten auszutauschen, ist das grundlegende Mittel, um verteilte Algorithmen für kooperative Netzwerke ohne Fusionszentrum zu entwerfen. Dieses Schlüsselverfahren wurde durch das kollektive Verhalten diverser Tiergruppen, wie beispielsweise Bienenschwärme, Bakterienkolonien, Staren- und Fischschwärme, inspiriert. In diesen und vielen anderen Fällen weist die Gruppe als Ganzes ein Verhalten auf, das an den Individuen selbst nicht beobachtet werden kann. Diesem organisierten Verhalten kann auf den Grund gegangen werden, wenn man die Vielzahl an Interaktionen zwischen den Agenten betrachtet.

In vielen Netzwerkanwendungen ist es erforderlich, verschiedene, in der Umgebung präsente Modelle zu erkennen und zu verfolgen. In unserer Forschung konzentrieren wird uns auf Netzwerke mit mehreren Aufgaben, in denen die einzelnen Agenten Interesse an unterschiedlichen Zielen haben können. Eine Schwierigkeit in solchen Netzwerken ist die Tatsache, dass die Agenten im Vorhinein nicht wissen, welche Modelle von ihren Nachbarn beobachtet werden. Außerdem ist ihnen die Gesamtzahl der beobachteten Modelle sowie deren Indizes nicht bekannt. Wir schlagen ein adaptives und verteiltes Gruppierungsverfahren vor, das es den Agenten ermöglicht, von eingehenden Datenströmen zu lernen und darauf basierend auf robuste Weise zu gruppieren. Sobald Gruppen gebildet wurden, kann die Kooperation zwischen den Agenten mit gleichem Ziel die Leistung des Schätzverfahrens steigern. Basierend auf der Gruppenbildung können die ungenutzten Verbindungen zwischen Agenten mit unterschiedlichen Zielen genutzt werden, um Agenten mit demselben Ziel aber ohne direkte Verbindung zu verknüpfen. Wir analysieren die Leistung des Gruppierungsprozesses and zeigen, dass die Fehlerwahrscheinlichkeiten bei der Gruppenbildung exponentiell zu Null abfallen. Ferner untersuchen wir die mittlere quadratische Leistung des vorgestellten Gruppierungsverfahrens und schlagen ein verteiltes Kennzeichnungssystem vor, das jeder Gruppe einen eindeutigen Index für ihr beobachtetes Modell zuweist.

Bestimmte Tiergruppen, wie beispielsweise Bienenschwärme, bestehen aus informierten und uninformierten Agenten und nur erstere sammeln Informationen über die Umgebung. Wir betrachten ein Netzwerk, in dem die informierten Agenten unterschiedliche Modelle beobachten und Informationen über Ihre Beobachtungen an die uninformierten Agenten übermitteln. Jeder uninformierte Agent antwortet einem informierten und tritt

dessen Gruppe bei. Wir schlagen ein adaptives und verteiltes Gruppierungs- und Partitionsverfahren vor, das die informierten Agenten basierend auf ihren beobachteten Modellen in verschiedene Gruppen einteilt. Anschließend wenden wir eine verteilte Strategie an, um die uninformierten Agenten in etwa gleich große Gruppen um die informierten Gruppen herum zu verteilen.

In manch anderen Situationen müssen sich die Agenten zwischen mehreren Optionen, wie z.B. welche Nahrungsquelle sie verfolgen sollen, entscheiden. Wir stellen ein Verfahren zur verteilten Entscheidungsfindung in adaptiven Netzwerken vor, in dem die Agenten Daten von verschiedenen Modellen sammeln. Die Agenten müssen sich für ein Modell entscheiden, das sie schätzen und verfolgen. Sobald sich das Netzwerk auf ein gewünschtes Modell geeinigt hat, verbessert die Kooperation zwischen den Agenten die Schätzleistung, indem Daten über das Netzwerk weitergeleitet werden.

Wir untersuchen alle Szenarien und Methoden sowohl in statischen als auch in mobilen Netzwerken. Die Simulationen veranschaulichen die Leistung der vorgestellten Strategien und vergleichen diese mit dem aktuellen Stand der Forschung.

Abstract

This PhD thesis focuses on cooperative multi-task networks. Cooperative networks consist of a collection of agents with adaptation and learning abilities. The idea of sharing data among the neighboring agents is the basic tool for designing distributed algorithms for cooperative networks without a fusion center. This key technique is inspired by the collective behavior of some animal groups such as bee swarms, bacteria colonies, starling flocks, and fish schools. In these cases and in many more, the group of individuals as a whole exhibits a behavior that cannot be accessed at the individual members. This organized behavior can be understood by considering the large amount of interactions among agents.

There arises the need in many network applications to infer and track different models of interest in an environment. In our research, we focus on multi-task networks where the individual agents might be interested in different objectives. One challenge in these networks is that the agents do not know beforehand which models are being observed by their neighbors. Furthermore, the total number of the observed models and their indices are not available to them, either. We propose a distributed clustering technique that allows the agents to learn and form their clusters from streaming data in a robust manner. Once clusters have been formed, cooperation among agents with similar objectives can increase the performance of the inference task. Based on the cluster formation, the unused links among the agents that track different models are exploited to link the agents that are interested in the same model but do not have direct links between each other. We analyze the performance of the clustering scheme and show that the clustering error probabilities decay exponentially to zero. In addition, we examine the mean-square performance of the proposed clustering scheme. Furthermore, we propose a distributed labeling system, which ensures that each cluster has a unique index for its observed model.

Certain types of animal groups, such as bee swarms, consist of informed and uninformed agents where only the informed agents collect information about the environment. We consider a network where the informed agents observe different models and send information about them to the uninformed ones. Each uninformed agent responds to one informed agent and joins its group. We suggest an adaptive and distributed clustering and partitioning approach that allows the informed agents in the network to be clustered into different groups according to the observed models; then we apply a decentralized strategy to split the uninformed agents into groups of approximately equal size around the informed agents.

In some other situations, the agents in the network need to decide between multiple options, for example, to track only one of multiple food sources. We propose a distributed decision-making approach over adaptive networks where agents in the network collect data generated by different models. The agents need to decide which model to estimate and track. Once the network reaches an agreement on one desired model, the cooperation among the agents enhances the performance of the estimation task by relaying data throughout the network.

We investigate all scenarios and approaches in both cases: static and mobile networks. The simulations illustrate the performance of the proposed strategies and compare them with state-of-the-art approaches.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

'The beginning is the most important part of the work.'

Plato

Lately, biological multi-agent systems have been adopted as a source of inspiration for self-organizing systems. Self-organizing has been studied in biology showing a rich variety of collaborative behaviors and presenting interesting characteristic such as scalability and fault tolerance [1]. One example of these self-organizing systems are fish swarms. A fish swarming system consists of a large number of homogenous, simple agents that interacting locally among themselves resulting in an interesting global behavior without any central control.

Apart from that, modern applications increasingly rely on small and wireless devices, such as mobile phones, laptops, and sensors, that interact each other. The steadily increasing number of these devices in daily life provides powerful basis. These emerging structures allow the implementation of a wide range of new application and services, such as rescue applications, disaster prevention, and monitoring applications. Some situations such as excluding or including agents in the network, time-varying network topologies, and unpredictable changes in the environment, cannot be handled by centralized design [1]. Bio-inspired algorithms are capable of providing low-cost and fast solutions to several problems [1–4]. The main idea of these algorithms is that the individual simple agents can overcome their individual limitations through collaboration to achieve complex tasks in a distributed manner.

Adaptive networks are suitable to model the complex and self-organized behavior of biological systems [5–7]. They consist of spatially distributed agents that are linked together through a topology and cooperate with each other through local interactions to solve distributed inference problems in real-time. Two classes of fully decentralized strategies have been studied in the literature, namely, consensus and diffusion strategies. The consensus strategies were proposed in the context of distributed optimization and estimation problems [8]. While diffusion strategies were introduced to solve distributed estimation and adaptation problems [9–16].

In this thesis, decentralized approaches that mimic and simulate selective interesting collective behaviors of animal groups are considered. We explain each approach with

the data and network model. We propose some error measurements for evaluation. Moreover, we provide some theoretical performance analysis of the proposed algorithms and validate them with practical simulations.

1.1 Contributions

- **Decentralized Clustering and Linking by Networked Agents:**

  We consider the problem of decentralized clustering and estimation over multi-task networks, where the agents infer and track different models of interest. The agents do not know beforehand which model is generating the data they sense. They also do not know which agents in their neighborhood belong to the same cluster. We propose a decentralized clustering algorithm aimed at identifying and forming clusters of agents of similar objectives, and at guiding cooperation to enhance the inference performance. While links between agents following different objectives are ignored in the clustering process, we show how to exploit these links to relay critical information across the network for enhanced performance [17–19]. Moreover, the agents do not know the index of their observed models. We propose a labeling system that depends on an electing process to elect a master agent for each cluster. This master agent provides a label for its cluster, thus ensuring that each cluster has a unique model index.

- **Decentralized Partitioning Over Inhomogeneous Multi-Agent Networks:**

  We propose a decentralized partitioning technique aimed at implementing a dynamic multi-task network using adaptation and learning in the presence of informed and uninformed agents. The algorithm ensures a fair partitioning process to distribute the uninformed agents among the groups that are interested in several objectives. Moreover, the decentralized technique has a self-organizing feature that endows the network with a learning ability in stationary and non-stationary environments. We have applied the proposed technique in both static and mobile networks and shown that the size of the groups matches the centralized partitioning size well [20].

- **Decentralized Decision-Making Over Adaptive Networks:**

  We consider a distributed mean-square-error estimation problem over a multi-task network. Two definitions are introduced: the observed model, i.e., to the one, from which an agent collects data, and the desired model, i.e., the one towards which the agent decides to move. The agents do not know which model generated the

data they collect; they also do not know which other agents in their neighborhood sense data arising from the same model. Therefore, each agent needs to determine the subset of its neighbors that observes the same model. The objective is to reach an agreement among all agents in the network on one common model to estimate and track [21].

1.2 Publications

The period of doctoral candidacy has culminated in the following publications:

Internationally Refereed Journal Articles

- S. Khawatmi, A. H. Sayed, and A. M. Zoubir, "Decentralized decision-making over multi-task networks," (under review), 2017.

- S. Khawatmi, A. H. Sayed, and A. M. Zoubir, "Decentralized clustering and linking by networked agents," *IEEE Trans. Signal Processing*, vol. 65, pp. 3526–3537, July 2017.

Internationally Refereed Conference Papers

- S. Khawatmi, X. Huang, and A. M. Zoubir, "Distributed decision-making over mobile adaptive networks," *in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (New Orleans, USA), March 2017, pp. 3864–3868.

- S. Khawatmi and A. M. Zoubir, "Decentralized partitioning over adaptive networks," *in Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, (Vietri sul Mare, Salerno, Italy), September 2016, pp. 1–6.

- S. Khawatmi, A. M. Zoubir, and A. H. Sayed, "Decentralized clustering over adaptive networks," *in Proc. 23rd European Signal Processing Conference (EUSIPCO)*, (Nice, France), September 2015, pp. 2745–2749.

- S. Khawatmi, "Signal processing over multi-task networks," *in Proc. 23rd European Signal Processing Conference (EUSIPCO)*, (Nice, France), September 2015, p. 1562.

1.3 Dissertation Overview

Following this introduction, the dissertation is organized as follows:

An introduction to learning and optimization over multi-agent networks relating to state-of-the-art in this field is considered in **Chapter 2**.

Chapter 3 presents decentralized clustering and linking algorithms by networked agents. We first explain the algorithms. Then, the performance analysis and the simulation results are provided. Finally, we propose an approach to label models over multi-task networks.

In **Chapter 4** a partitioning approach over multi-task networks is considered. We demonstrate the partitioning procedure and the simulation results for both static and mobile networks.

Chapter 5 describes the third contribution, decentralized decision-making over multi-task networks. A decision-making technique that deals with multiple models over networks is proposed. We illustrate the motion scheme of the mobile networks with the decision-making technique. Then, we explore the convergence of the process of achieving the agreement over the network theoretically and practically.

Finally, **Chapter 6** concludes and summarizes the thesis.

Chapter 2

Fundamentals and Related Work

'If I have seen further it is by standing on the shoulders of giants.'

Isaac Newton

In this chapter we review the optimization problem of agents over networks for both single-task and multi-task networks [7, 22–25]. We present the gradient-descent algorithm, which is commonly used for adaptation, learning, and optimization problems by stand-alone single agents and networked agents. The case of real-valued arguments and constant step-sizes is considered. In addition, we explain the motion mechanism for mobile adaptive networks. Finally, the state-of-the-art in the field of multi-task networks is highlighted[1].

Notation:
We use lowercase letters to denote vectors, uppercase letters for matrices, plain letters for deterministic variables, and boldface letters for random variables. The superscript $\circ$ is used to indicate true values. The letter $\mathbb{E}$ denotes the expectation operator. The Euclidean norm is denoted by $\|\cdot\|$. The symbols $\mathbb{1}$ and I denote the all-one vector and the identity matrix of appropriate sizes, respectively. We write $(\cdot)^\intercal$, $(\cdot)^{-1}$, and $\mathrm{Tr}(\cdot)$ to denote transposition, matrix inversion, and matrix trace, respectively. The $\mathrm{diag}(\cdot)$ operator extracts the diagonal entries of its matrix argument and stacks them into a column.

2.1 Risk and Loss Functions

Let $J(w) \in \mathbb{R}$ denote a real-valued cost function of a real-valued vector argument, $w \in \mathbb{R}^M$. The cost function $J(w)$ is the expectation of some loss function $Q(w; \boldsymbol{x})$:

$$J(w) = \mathbb{E}\, Q(w; \boldsymbol{x}). \tag{2.1}$$

[1]This chapter presents the fundamentals of adaptation, learning, and optimization over networks mostly based on the book [22].

We denote the gradient vectors of $J(w)$ relative to w and $w^\intercal$ by the following vectors:

$$\nabla_w J(w) \triangleq \left[\frac{\partial J(w)}{\partial w_1} \; \frac{\partial J(w)}{\partial w_2} \; \cdots \; \frac{\partial J(w)}{\partial w_M}\right], \tag{2.2}$$

$$\nabla_{w^\intercal} J(w) \triangleq \big[\nabla_w J(w)\big]^\intercal, \tag{2.3}$$

respectively, where $\nabla_w J(w)$ is a row vector, while $\nabla_{w^\intercal} J(w)$ is a column vector. The individual entries of w are given by

$$w \triangleq \mathrm{col}\{w_1, w_2, \ldots, w_M\}. \tag{2.4}$$

We define the Hessian matrix of $J(w)$ with respect to w as follows:

$$\nabla_w^2 J(w) \triangleq \nabla_{w^\intercal} J(w)\big[\nabla_w J(w)\big] = \nabla_w J(w)\big[\nabla_{w^\intercal} J(w)\big] \tag{2.5}$$

where $\nabla_w^2 J(w)$ is an $M \times M$ symmetric matrix. Quadratic costs are widely used in estimation and adaptation problems.

Mean-square-error cost [22]:
Let $\boldsymbol{d}$ denote a zero-mean scalar random variable with variance $\sigma_d^2 = \mathbb{E}\,\boldsymbol{d}^2$ and let $\boldsymbol{u}$ denote a zero-mean $1 \times M$ random vector with covariance matrix $R_u = \mathbb{E}\,\boldsymbol{u}^\intercal \boldsymbol{u} > 0$. The variables $\{\boldsymbol{d}, \boldsymbol{u}\}$ represent the random variable $\boldsymbol{x}$ mentioned in (2.1). The cross covariance vector is denoted by $r_{du} = \mathbb{E}\,\boldsymbol{d}\boldsymbol{u}^\intercal$. The idea is to seek the vector w° that minimizes the following cost function:

$$J(w) \triangleq \mathbb{E}\,(\boldsymbol{d} - \boldsymbol{u}w)^2 = \sigma_d^2 - 2r_{du}^\intercal w + w^\intercal R_u w. \tag{2.6}$$

This quadratic cost function corresponds to the following loss function:

$$Q(w; \boldsymbol{x}) \triangleq (\boldsymbol{d} - \boldsymbol{u}w)^2 = \boldsymbol{d}^2 - 2\boldsymbol{d}\boldsymbol{u}w + w^\intercal \boldsymbol{u}^\intercal \boldsymbol{u} w. \tag{2.7}$$

The gradient vector and Hessian matrix of $J(w)$ are given by

$$\nabla_w J(w) = 2(R_u w - r_{du})^\intercal, \tag{2.8}$$

$$\nabla_w^2 J(w) = 2R_u. \tag{2.9}$$

Figure 2.1 shows an illustration of the mean-square-error cost for the two-dimensional case, where $M = 2$. The individual entries of $w \in \mathbb{R}^M$ are given by $w = \mathrm{col}\{w_1, w_2\}$.

2.2 Optimization via Gradient-Descent

Stochastic gradient algorithms are powerful iterative methods for solving optimization problems of the following form:

$$w^\circ = \arg\min_w J(w). \tag{2.10}$$

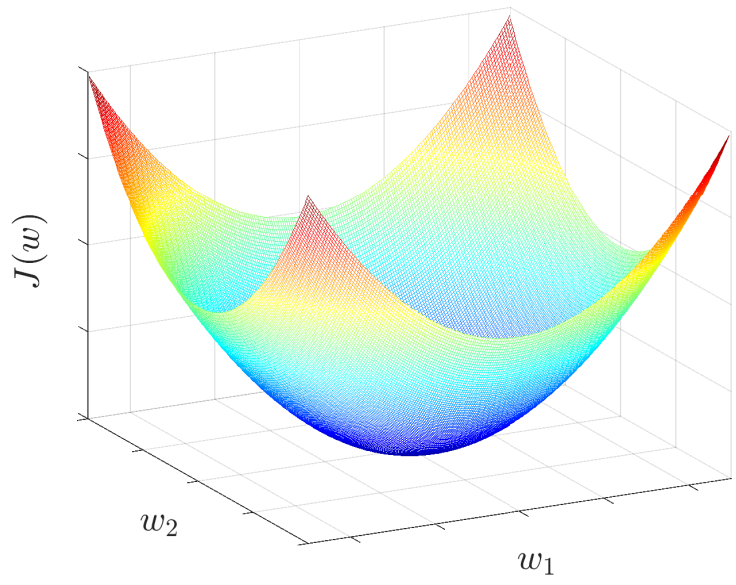

Figure 2.1. Quadratic cost function for the two-dimensional case, $M = 2$.

To proceed in solving the optimization problem we assume the following conditions on the cost function:

1. The cost function $J(w)$ is twice–differentiable.
2. The cost function $J(w)$ is ν–strongly convex and satisfies – for some positive parameters $\nu \leq \delta$:
$$0 < \nu I_M \leq \nabla_w^2 J(w) \leq \delta I_M. \tag{2.11}$$
3. The gradient vector of $J(w)$ is required to be δ–Lipschitz:
$$\|\nabla_w J(w_2) - \nabla_w J(w_1)\| \leq \delta \|w_2 - w_1\|. \tag{2.12}$$

In this thesis we focus on the gradient-descent algorithm, which requires knowledge of the actual gradient vector and has the following form for a constant step-size $\mu > 0$:

$$w_i = w_{i-1} - \mu \nabla_{w^\intercal} J(w_{i-1}) \tag{2.13}$$

where $i \geq 0$ is the index of time (iteration). Assume that the cost function $J(w)$ satisfies the conditions (2.11) and (2.12). It then holds that for any initial iterate w_{-1} the gradient-descent algorithm (2.13) generates iterates w_i that converge exponentially fast to the global minimizer, w°, if the step-size μ satisfies

$$0 < \mu < \frac{2\nu}{\delta^2}. \tag{2.14}$$

We denote the error vector at time instant i by $\widetilde{w}_i$ and calculate it as

$$\widetilde{w}_i = w^\circ - w_i. \tag{2.15}$$

We can write

$$\|\widetilde{w}_i\|^2 \leq \alpha \|\widetilde{w}_{i-1}\|^2 \tag{2.16}$$

where $0 \leq \alpha < 1$ is a real-valued scalar and given by

$$\alpha = 1 - 2\mu\nu + \mu^2\delta^2. \tag{2.17}$$

Mean-square-error cost [22]:
For the quadratic cost example, the Hessian matrices in the example of the mean-square-error cost satisfy (2.11), where

$$2\lambda_{\min}(R_u) I_M \leq \nabla_w^2 J(w) \leq 2\lambda_{\max}(R_u) I_M \tag{2.18}$$

where $\lambda_{\min}(R_u)$ ($\lambda_{\max}(R_u)$) is the minimum (maximum) eigenvalue of the matrix R_u. This implies that: $\delta = 2\lambda_{\max}(R_u)$ and $\nu = 2\lambda_{\min}(R_u)$. By setting the gradient vector in (2.8) to zero, the minimizer w° is given by the unique solution to the equations $R_u w^\circ = r_{du}$. Substituting (2.8) to the gradient vector into the step (2.13), leads to the following iterative algorithm

$$w_i = w_{i-1} - 2\mu(r_{du} - R_u w_{i-1}). \tag{2.19}$$

The iterates w_i generated by this recursion will converge to w° at an exponential rate for any step-size satisfying condition (2.14), which is given by

$$\mu < \frac{\lambda_{\min}(R_u)}{\lambda_{\max}^2(R_u)}. \tag{2.20}$$

2.2.1 Stochastic Gradient-Descent

As mentioned before, the gradient-descent algorithm requires knowledge of the actual gradient vector of the cost function. Since in the context of adaptation and learning this information is not available beforehand, we need to approximate it. We do so by replacing the true gradient with an approximate gradient leading to stochastic gradient algorithms. This approximation causes a gradient error that interferes with the operation of the algorithm. Therefore, it is important to evaluate resulting performance degradation. Jointly, the stochastic approximation step provides a tracking property into the operation of the gradient-descent algorithm. The stochastic gradient-descent

is able to track drifts in the minimizer location. This is because stochastic gradient implementations approximate the gradient vector from streaming data and, therefore, the drifts in the signal models will be reflected in these realizations.

Let $J(w) \in \mathbb{R}$ denote the real-valued cost function of a real-valued vector argument $w \in \mathbb{R}^M$ and consider the following minimization problem:

$$w^\circ = \arg\min_w J(w). \tag{2.21}$$

We assume that $J(w)$ is twice-differentiable and satisfies conditions (2.11) and (2.12). Again, the cost function $J(w)$ is the expectation of some loss function $Q(w; \boldsymbol{x}_i)$ and is given by

$$J(w) = \mathbb{E}\, Q(w; \boldsymbol{x}_i) \tag{2.22}$$

where the expectation is taken over the distribution of $\boldsymbol{x}_i$. The traditional gradient-descent algorithm for solving (2.22) was described by (2.13) as follows:

$$w_i = w_{i-1} - \mu \nabla_{w^\intercal} J(w_{i-1}). \tag{2.23}$$

The true gradient vector, $\nabla_{w^\intercal} J(w_{i-1})$, in (2.23) is not available. The exact form of $J(w)$ is also unknown since the expectation of $Q(w; \boldsymbol{x}_i)$ cannot be computed. Therefore, we replace the true gradient vector $\nabla_{w^\intercal} J(w_{i-1})$ by an instantaneous approximation denoted by $\widehat{\nabla_{w^\intercal} J}(w_{i-1})$. This replacement leads to the following stochastic-gradient recursion:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - \mu \widehat{\nabla_{w^\intercal} J}(\boldsymbol{w}_{i-1}). \tag{2.24}$$

Mean-square-error cost (LMS) [22]:
Let $\boldsymbol{d}(i)$ denote a streaming sequence of zero-mean random variables with variance $\sigma_d^2 = \mathbb{E}\, \boldsymbol{d}(i)^2$. Let $\boldsymbol{u}_i$ denote a streaming sequence of $1 \times M$ independent zero-mean random vectors with covariance matrix $R_u = \mathbb{E}\, \boldsymbol{u}_i^\intercal \boldsymbol{u}_i > 0$. Both processes $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ are assumed to be jointly wide-sense stationary. The cross-covariance vector of $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ is given by $r_{du} = \mathbb{E}\, \boldsymbol{d}(i)\boldsymbol{u}_i^\intercal$. The processes $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ are related via a linear regression model of the form

$$\boldsymbol{d}(i) = \boldsymbol{u}_i w^\circ + \boldsymbol{v}(i) \tag{2.25}$$

for some unknown vector w°. We assume that the noise process $\boldsymbol{v}(i)$ is zero-mean white with power $\sigma_v^2 = \mathbb{E}\, \boldsymbol{v}(i)^2$ and independent of $\boldsymbol{u}_j$ for all i, j. We seek to estimate w° by minimizing the following mean-square error cost:

$$J(w) = \mathbb{E}\, (\boldsymbol{d}(i) - \boldsymbol{u}_i w)^2 \tag{2.26}$$

where the processes $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ represent the random data $\boldsymbol{x}_i$ in (2.24). The gradient-descent recursion that is given by (2.19) requires knowledge of the second-order moments R_u and r_{du}. This information is not available beforehand. Instead, the adaptive

agent senses realizations $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$, the statistical distributions of which have moments $\{r_{du}, R_u\}$. We can use these realizations to approximate the moments and the true gradient vector. Different constructions for this purpose lead to different adaptive algorithms. One option is to use the data $\{\boldsymbol{d}(i), \boldsymbol{u}_i\}$ to compute instantaneous approximations for the unavailable moments at every time instant i as follows:

$$r_{du} \approx \boldsymbol{d}(i)\boldsymbol{u}_i^{\mathsf{T}}, \quad R_u \approx \boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{u}_i. \tag{2.27}$$

The true gradient vector is approximated by:

$$\widehat{\nabla_{w^{\mathsf{T}}} J}(w) = 2(\boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{u}_i w - \boldsymbol{u}_i^{\mathsf{T}}\boldsymbol{d}(i)). \tag{2.28}$$

Using the approximations in (2.27) leads to the following least-mean-squares (LMS) algorithm:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - 2\mu\, \boldsymbol{u}_i^{\mathsf{T}}(\boldsymbol{d}(i) - \boldsymbol{u}_i\boldsymbol{w}_{i-1}). \tag{2.29}$$

2.2.2 Gradient Noise Process

The approximate gradient vector in (2.24) introduces disturbances that are not present in the original recursion in (2.23). We refer to the perturbation as gradient noise, which is defined as

$$\boldsymbol{s}_i(\boldsymbol{w}_{i-1}) \triangleq \widehat{\nabla J}_{w^{\mathsf{T}}}(\boldsymbol{w}_{i-1}) - \nabla J_{w^{\mathsf{T}}}(\boldsymbol{w}_{i-1}). \tag{2.30}$$

In the presence of the noise disturbance $\boldsymbol{s}_i(\boldsymbol{w}_{i-1})$ the stochastic iterate $\boldsymbol{w}_i$ will not converge to the minimizer w° using constant step-sizes. Some decay in performance occurs because the iterate $\boldsymbol{w}_i$ will fluctuate close to w° in steady-state.

Let the symbol $\mathbb{F}_{i-1}$ represent the collection of all possible random events generated by the past iterates $\{\boldsymbol{w}_j\}$ up to time $(j \leq i-1)$. The filtration $\mathbb{F}_{i-1}$ represents the information about $\{\boldsymbol{w}_j\}$ that is available up to time $(i-1)$,

$$\mathbb{F}_{i-1} = \text{filtration}\{\boldsymbol{w}_{-1}, \boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}\}. \tag{2.31}$$

Some conditions on the stochastic property of the gradient noise process (2.30) are assumed in order to examine the convergence and performance of the stochastic-gradient recursion in (2.24).

It is assumed that the first and second-order conditional moments of the gradient noise process satisfy the following conditions for any $\boldsymbol{w} \in \mathbb{F}_{i-1}$:

$$\mathbb{E}\ [\boldsymbol{s}_i(\boldsymbol{w})\ |\mathbb{F}_{i-1}] = 0 \tag{2.32}$$

$$\mathbb{E}\ \left[\|\boldsymbol{s}_i(\boldsymbol{w})\|^2\ |\mathbb{F}_{i-1}\right] \leq \bar{\beta}^2\|\boldsymbol{w}\|^2 + \bar{\sigma}_s^2 \tag{2.33}$$

for some non-negative scalars $\bar{\beta}^2$ and $\bar{\sigma}_s^2$. Conditions (2.32) and (2.33) imply that

$$\mathbb{E}\ [\boldsymbol{s}_i(\boldsymbol{w}_{i-1})\ |\mathbb{F}_{i-1}] = 0 \tag{2.34}$$

$$\mathbb{E}\ \left[\|\boldsymbol{s}_i(\boldsymbol{w}_{i-1})\|^2\ |\mathbb{F}_{i-1}\right] \leq \beta^2\|\widetilde{\boldsymbol{w}}_{i-1}\|^2 + \sigma_s^2 \tag{2.35}$$

where β^2 and σ_s^2 are non-negative scalars given by

$$\beta^2 = 2\bar{\beta}^2 \tag{2.36}$$

$$\sigma_s^2 = 2\bar{\beta}^2\|w^\circ\|^2 + \bar{\sigma}_s^2. \tag{2.37}$$

The error vector $\widetilde{\boldsymbol{w}}_{i-1}$ is given by

$$\widetilde{\boldsymbol{w}}_{i-1} = w^\circ - \boldsymbol{w}_{i-1}. \tag{2.38}$$

Taking the expectation of (2.34) and (2.35) implies that the gradient noise process satisfies

$$\mathbb{E}\ \boldsymbol{s}_i(\boldsymbol{w}_{i-1}) = 0 \tag{2.39}$$

$$\mathbb{E}\ \|\boldsymbol{s}_i(\boldsymbol{w}_{i-1})\|^2 \leq \beta^2\ \mathbb{E}\ \|\widetilde{\boldsymbol{w}}_{i-1}\|^2 + \sigma_s^2. \tag{2.40}$$

2.2.3 Stability of First, Second, and Fourth-Order Error Moments

In this section we study the convergence of the stochastic-gradient recursion from (2.24) in the mean-square-error sense.

2.2.3.1 Mean-Square Stability

Assume that the conditions (2.11), (2.12), (2.32), and (2.33) on the cost function and the gradient noise process hold. Consider the non-negative scalars in (2.36) and (2.37). For any step-size value, μ, satisfying

$$\mu < \frac{2\nu}{\delta^2 + \beta^2} \tag{2.41}$$

it holds that $\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^2$ converges exponentially and for sufficiently small step-sizes it holds that

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{w}}_i\|^2 = \mathcal{O}(\mu). \tag{2.42}$$

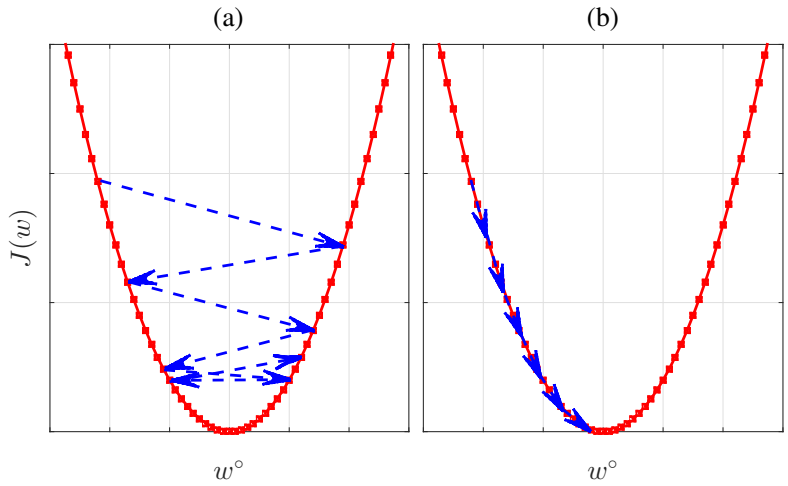

Figure 2.2. Illustration of the impact of the step-size for two cases: μ is not sufficiently small (a) and μ is sufficiently small (b).

2.2.3.2 Fourth-Order Moment Stability

To establish the convergence of the fourth-order moment $\mathbb{E}\|\widetilde{\boldsymbol{w}}_i\|^4$ to a bounded region. It is assumed that the first and fourth-order conditional moments of the gradient noise process satisfy the following conditions for any $\boldsymbol{w} \in \mathbb{F}_{i-1}$:

$$\mathbb{E}\ [\boldsymbol{s}_i(\boldsymbol{w})\ |\mathbb{F}_{i-1}] = 0 \tag{2.43}$$

$$\mathbb{E}\ \left[\|\boldsymbol{s}_i(\boldsymbol{w})\|^4\ |\mathbb{F}_{i-1}\right] \leq \bar{\beta}^2\|\boldsymbol{w}\|^4 + \bar{\sigma}_s^4 \tag{2.44}$$

for some non-negative scalars $\bar{\beta}^2$ and $\bar{\sigma}_s^2$.

Assume the conditions (2.11), (2.12), (2.43), and (2.44) on the cost function and the gradient noise process hold, for sufficiently small step-sizes it holds that

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{w}}_i\|^2 = \mathcal{O}(\mu) \tag{2.45}$$

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{w}}_i\|^4 = \mathcal{O}(\mu^2). \tag{2.46}$$

In the following example we will explain the impact of step-size μ on the learning process. Let $M = 1$ as shown in Fig. 2.2. Using a step-size μ that is not sufficiently small leads to the case (a) where it is not possible to hit the minimum value w° of the cost function $J(w)$ with a small error. However, choosing step-size μ sufficiently small allows the gradient-decent process to converge to the minimum value w° of the cost function $J(w)$ with a small error as shown in case (b).

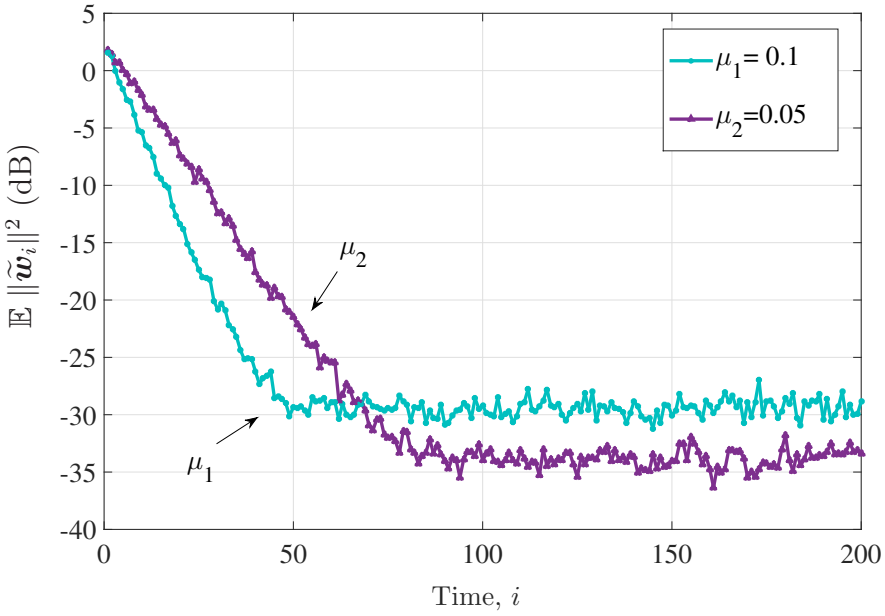

Figure 2.3. Simulated second-order error moment, $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$, using two different step-sizes, μ_1 and μ_2.

Figure 2.3 depicts the simulated second-order error moment $\mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2$ for two cases: $\mu_1 = 0.1$ and $\mu_2 = 0.05$. With a large step-size the error curve converges faster but with a higher steady-state error. However, with a sufficiently small step-size the error curve converges slower but reaches a lower level in steady-state.

The term *steady-state* refers to the operation of the stochastic-gradient implementation after sufficient iterations have elapsed, i.e., as $i \to \infty$.

2.2.3.3 Mean Stability

To establish the convergence of the first-order moment $\|\mathbb{E}\,\widetilde{\boldsymbol{w}}_i\|$ to a bounded region, we assume the following smoothness conditions: It is assumed that the Hessian matrix of the cost function, $J(w)$, and the noise covariance matrix defined by

$$R_{s,i}(\boldsymbol{w}) \triangleq \mathbb{E}\left[\boldsymbol{s}_i(\boldsymbol{w})\boldsymbol{s}_i^{\mathsf{T}}(\boldsymbol{w}) \mid \mathbb{F}_{i-1}\right] \tag{2.47}$$

are locally Lipschitz continuous in a small region around $w = w^\circ$, i.e.,

$$\|\nabla_w^2 J(w^\circ + \Delta w) - \nabla_w^2 J(w^\circ)\| \leq \kappa_1 \|\Delta w\| \tag{2.48}$$

$$\|R_{s,i}(w^\circ + \Delta w) - R_{s,i}(w^\circ)\| \leq \kappa_2 \|\Delta w\|^\gamma \tag{2.49}$$

for a small $\|\Delta w\| \leq \epsilon$ and for some constants $\kappa_1 \geq 0$, $\kappa_2 \geq 0$, and $0 < \gamma \leq 4$.

Now, assume the conditions (2.11), (2.12), (2.43), (2.44), and (2.48) on the cost function and the gradient noise process hold. Then, for sufficiently small step-sizes it holds that

$$\limsup_{i\to\infty} \|\mathbb{E}\, \widetilde{\boldsymbol{w}}_i\| = \mathcal{O}(\mu). \tag{2.50}$$

We assume that the covariance matrix in (2.47) approximates a constant value in the limit if we evaluate it at w°. This covariance matrix is given by

$$R_s \triangleq \lim_{i\to\infty} \mathbb{E}\, \big[\boldsymbol{s}_i(w^\circ)\boldsymbol{s}_i^{\mathsf{T}}(w^\circ) \mid \mathbb{F}_{i-1}\big]. \tag{2.51}$$

2.2.4 Long-Term Error Dynamics

Assume the conditions (2.11), (2.12), (2.43), (2.44), and (2.48) on the cost function and the gradient noise process hold. After sufficient iterations the error dynamics of the stochastic-gradient algorithm in (2.24) can be approximated by the following model:

$$\widetilde{\boldsymbol{w}}_i' = (I - \mu H)\widetilde{\boldsymbol{w}}_{i-1}' + \mu \boldsymbol{s}_i(\boldsymbol{w}_{i-1}) \tag{2.52}$$

where H is a constant, symmetric, and positive-definite matrix defined as the value of the Hessian matrix at the minimizer w°, i.e.,

$$H \triangleq \nabla_w^2 J(w^\circ). \tag{2.53}$$

It is easy to work with the recursion (2.52) because its dynamics are driven by the constant matrix H compared to the random matrix $\boldsymbol{H}_{i-1}$ in the original error recursion:

$$\widetilde{\boldsymbol{w}}_i = (I - \mu \boldsymbol{H}_{i-1})\widetilde{\boldsymbol{w}}_{i-1} + \mu \boldsymbol{s}_i(\boldsymbol{w}_{i-1}). \tag{2.54}$$

However, we will explore the mean-square-error expression of the long-term model (2.52) to provide an accurate representation of the mean-square-error of the original stochastic-gradient algorithm.

2.2.4.1 Mean-Square Stability of the Long-Term Model

Assume the conditions (2.11) and (2.12) on the cost function and the gradient noise process hold. For sufficiently small step-sizes μ, iterate $\boldsymbol{w}_i'$ which is generated by the long-term model (2.52), satisfies

$$\limsup_{i\to\infty} \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i'\|^2 = \mathcal{O}(\mu). \tag{2.55}$$

2.2.4.2 Mean Stability of the Long-Term Model

Assume the conditions (2.11), (2.12), (2.43), and (2.44) on the cost function and the gradient noise process hold. For sufficiently small step-sizes, iterate $\boldsymbol{w}_i'$ is asymptotically zero-mean, i.e.,

$$\lim_{i\to\infty} \mathbb{E}\, \widetilde{\boldsymbol{w}}_i' = 0. \tag{2.56}$$

2.2.4.3 Approximation Error

The approximation error refers to the difference between the original error recursion (2.54) and the long-term error recursion (2.52).

Assume the conditions (2.11), (2.12), (2.43), and (2.44) on the cost function and the gradient noise process hold. For sufficiently small step-sizes it holds that

$$\limsup_{i\to\infty} \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i - \widetilde{\boldsymbol{w}}_i'\|^2 = \mathcal{O}(\mu^2) \tag{2.57}$$

$$\limsup_{i\to\infty} \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i\|^2 = \limsup_{i\to\infty} \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i'\|^2 + \mathcal{O}(\mu^{3/2}). \tag{2.58}$$

2.2.5 Performance Metrics

2.2.5.1 Mean-Square Deviation (MSD)

The mean-square deviation measure is defined as follows:

$$\text{MSD} \triangleq \mu \left(\lim_{\mu\to 0} \limsup_{i\to\infty} \frac{1}{\mu}\, \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i\|^2 \right). \tag{2.59}$$

For sufficiently small step-sizes, we use the following definition for the mean-square deviation measure:

$$\text{MSD} \triangleq \lim_{i\to\infty} \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i\|^2. \tag{2.60}$$

2.2.5.2 Excess-Risk (ER)

We define the excess-risk (ER), which is called the excess-mean-square-error (EMSE) in the adaptive filtering literature [5, 7] as follows:

$$\text{ER} \triangleq \mu \left(\lim_{\mu \to 0} \limsup_{i \to \infty} \frac{1}{\mu} \mathbb{E} \left\{ J(\boldsymbol{w}_{i-1}) - J(w^\circ) \right\} \right). \tag{2.61}$$

Again, we write the definition for the ER more simply for sufficiently small step-sizes as

$$\text{ER} = \lim_{i \to \infty} \mathbb{E} \, \|\widetilde{\boldsymbol{w}}_{i-1}\|^2_{H/2} \tag{2.62}$$

where the operation $\|\cdot\|^2_\Sigma$ is the weighted squared Euclidean norm.

MSD and ER for the mean-square performance are given, by [22, pp. 391, 397]

$$\text{MSD} = \frac{\mu}{2} \text{Tr}(H^{-1} R_s) \tag{2.63}$$

$$\text{ER} = \frac{\mu}{4} \text{Tr}(R_s) \tag{2.64}$$

respectively, where H and R_s are defined as in (2.51) and (2.53).

Performance of LMS:
We have $H = 2R_u$ and $R_s = 4\sigma_v^2 R_u$ for the LMS recursion (2.29). For sufficiently small step-sizes, MSD and ER of the LMS filter are given by

$$\text{MSD} = \mu \, M \, \sigma_v^2 \tag{2.65}$$

$$\text{ER} = \mu \, \sigma_v^2 \, \text{Tr}(R_u) \tag{2.66}$$

respectively. Both of which are of order $\mathcal{O}(\mu)$.

2.3 Optimization in Stand-Alone Single Agents

2.3.1 Network Model

Figure 2.4 shows an example of a multi-agent network. We assume that the individual agents, $k = 1, 2, \ldots, N$, act without any cooperation among each other. The objective is to determine the unique minimizer w° of the aggregate cost

$$J^{\text{glob}}(w) \triangleq \sum_{k=1}^{N} J_k(w) \tag{2.67}$$

where each individual cost function $J_k(w)$ satisfies (2.11) and (2.12). In this non-cooperative case, each agent k runs the following stochastic-descent algorithm:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu \widehat{\nabla_{w^\intercal} J_k}(\boldsymbol{w}_{k,i-1}). \tag{2.68}$$

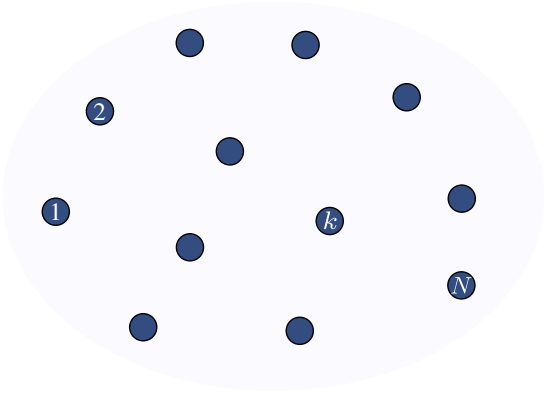

Figure 2.4. Network with stand-alone agents.

2.3.2 Stand-Alone LMS Networks

Each agent k receives the following streaming data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ at each time isntant i. We assume the data at each agent satisfies the statistical properties (2.11) and (2.12) and admits the linear regression model

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^\circ + \boldsymbol{v}_k(i) \tag{2.69}$$

where $k = 1, 2, \ldots, N$. We denote the statistical moments of the data at agent k by

$$\sigma_{v,k}^2 = \mathbb{E} \mid \boldsymbol{v}_k(i) \mid^2, \quad R_{u,k} = \mathbb{E}\, \boldsymbol{u}_{k,i}^\mathsf{T} \boldsymbol{u}_{k,i} > 0 \tag{2.70}$$

Assuming that all $R_{u,k}$ are equal across agents, i.e., $R_{u,k} = R_u$, the mean-square-error cost for each agent k is given by

$$J_k(w) = \mathbb{E} \mid \boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} w \mid^2 . \tag{2.71}$$

Each agent k satisfies the condition:

$$0 < \nu I_M \leq \nabla_w^2 J_k(w) \leq \delta I_M \tag{2.72}$$

with $\delta = 2\lambda_{\max}(R_u)$ and $\nu = 2\lambda_{\min}(R_u)$. Agents seek to estimate w° by running the LMS learning method which is described by the following recursion:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} - 2\mu \boldsymbol{u}_{k,i}^\mathsf{T} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{w}_{k,i-1}). \tag{2.73}$$

Agent k has an individual MSD performance that is given by

$$\mathrm{MSD}_{\mathrm{no},k} = \mu\, M\, \sigma_{v,k}^2. \tag{2.74}$$

The overall network MSD performance is given by:

$$\mathrm{MSD}_{\mathrm{no,net}} = \mu\, M \left(\frac{1}{N} \sum_{k=1}^{N} \sigma_{v,k}^2 \right). \tag{2.75}$$

Consequently, agents with a larger noise variance perform worse and have larger MSD than agents with a smaller noise variance.

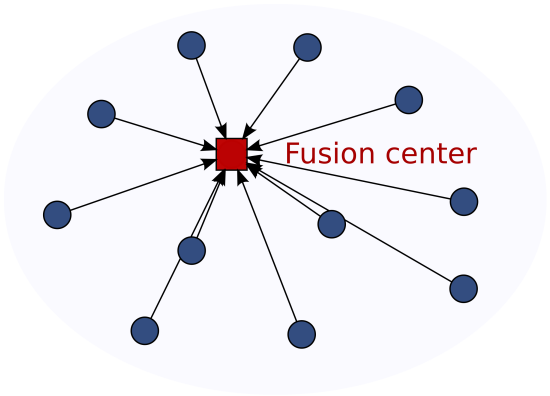

Figure 2.5. Network with a fusion center.

2.4 Optimization in Centralized Networks

2.4.1 Network Model

In this section we consider centralized networks. At every time instant i each agent transmits its data $\{\boldsymbol{d}_k(i), \boldsymbol{u}_{k,i}\}$ to a fusion center for processing as shown in Fig. 2.5. The objective is to seek the unique minimizer w° of the aggregate cost

$$J^{\text{glob}}(w) \triangleq \sum_{k=1}^{N} J_k(w). \tag{2.76}$$

In the centralized solution and under some conditions on the aggregate cost $J^{\text{glob}}(w)$ [22, p.414], it is sufficient for at least one of these costs $\{J_k(w)\}$ to be strongly-convex. while the remaining costs can be convex. This ensures the aggregate cost $J^{\text{glob}}(w)$ to be strong-convexity as well.

To seek the minimizer w°, the fusion center runs the following stochastic-gradient algorithm at each time instant i:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} - \frac{\mu}{N} \sum_{k=1}^{N} \widehat{\nabla_{w^{\mathsf{T}}} J_k}(\boldsymbol{w}_{i-1}). \tag{2.77}$$

2.4.2 Centralized LMS Networks

The fusion center runs a stochastic-gradient algorithm after receiving data from the agents as follows:

$$\boldsymbol{w}_i = \boldsymbol{w}_{i-1} + \mu \left(\frac{1}{N} \sum_{k=1}^{N} 2\boldsymbol{u}_{k,i}^{\mathsf{T}} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{w}_{i-1}) \right). \tag{2.78}$$

The MSD performance of the centralized LMS is given by,

$$\mathrm{MSD}_{\mathrm{cen}} = \mu\, M\, \frac{1}{N}\left(\frac{1}{N}\sum_{k=1}^{N}\sigma_{v,k}^2\right). \tag{2.79}$$

Note that $\mathrm{MSD}_{\mathrm{cen}}$ is proportional to $1/N$ times the average noise power across all agents in (2.74). Thus, it holds for sufficiently small step-sizes that

$$\frac{\mathrm{MSD}_{\mathrm{cen}}}{\mathrm{MSD}_{\mathrm{no,net}}} \leq \frac{1}{N}. \tag{2.80}$$

2.5 Optimization in Distributed Networks

Central control has several disadvantages such as the lack of robustness to fusion center failure, the lack of scalability, and the waste of communication resources. Distributed networks are useful in several contexts where scalability, robustness, and low power consumption are desirable. Here the agents do not need to have access to information from all other agents. Instead, they only interact with a limited number of neighbors. This limited cooperation is sufficient to attain a performance that is comparable to that of the centralized scheme [26–30].

Distributed optimization and estimation over networks without a fusion center has received interest in recent years, culminating in diffusion adaptation strategies and incremental adaptive strategies [9–12, 14, 15, 31–33].

2.5.1 Network Model

Consider a connected network of N agents. The network is represented by a graph consisting of N nodes and a set of edges connecting the nodes to each other. These nodes are labeled by $k = 1, 2, \ldots, N$. An edge that connects a node to itself is called a self-loop. Nodes connected by edges are called neighbors. The neighborhood of agent k is denoted by $\mathcal{N}_k$ and it consists of all agents that are connected to k by an edge, including k itself. We introduce the $N \times N$ adjacency matrix $Y = [y_{\ell k}]$, whose elements are either zero or one depending on whether the agents are linked by an edge or not as depicted in Fig 2.6. Specifically,

$$y_{\ell k} = \begin{cases} 1, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.81}$$

Agents that are linked by an edge can share information. For emphasis, in Fig 2.6 we represent the edges between agents k and ℓ by two separate directed arrows with $y_{\ell k}$ and $y_{k\ell}$.

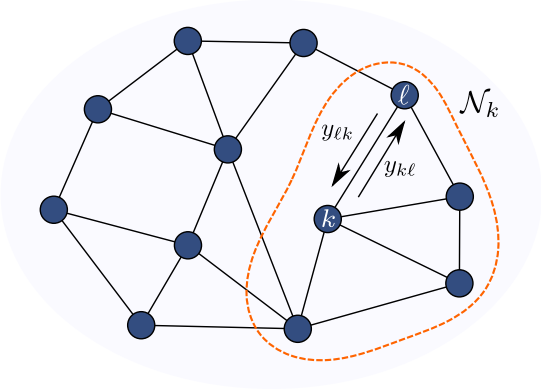

Figure 2.6. Distributed network without a fusion center. The neighborhood of agent k is denoted by $\mathcal{N}_k$.

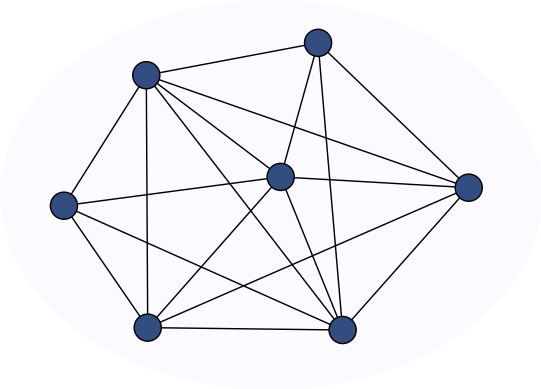

Figure 2.7. A fully-connected network.

2.5.1.1 Fully-Connected Networks

Figure 2.7 shows an example of a fully-connected network where the size of each neighborhood is equal to the network size, i.e., $n_k = N$, where $k = 1, 2, \ldots N$. Each agent k has access to the data from all other agents. Therefore, each agent k runs the following centralized stochastic-gradient step:

$$\boldsymbol{w}_{k,i} = \boldsymbol{w}_{k,i-1} - \frac{\mu}{N} \sum_{\ell=1}^{N} \widehat{\nabla_{w^\intercal} J_\ell}(\boldsymbol{w}_{k,i-1}). \tag{2.82}$$

2.5.1.2 Combination Matrix

Consider an $N \times N$ matrix A with non-negative real entries $a_{\ell k}$ satisfying

$$a_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k, \quad \mathbb{1}^\intercal A = \mathbb{1}^\intercal. \tag{2.83}$$

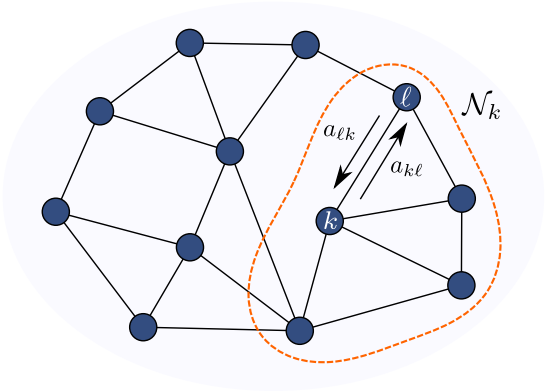

Figure 2.8. Distributed network where the weights $\{a_{\ell k}, a_{k\ell}\}$ represent the amount of data that can flow through the links between agents.

Figure 2.8 shows the scale of the data exchange between agents k and ℓ. There are several ways by which the combination weights can be selected. We can see that the combination weights are dictated by the sizes of the neighborhoods (or by the degrees of the agents). Note that when the neighborhoods vary with time, the degrees will also vary. The following rules are some of the popular ones and satisfy condition (2.83), for $\ell \in \mathcal{N}_k$ [9]:

1. **Averaging rule (uniform):** generates a left-stochastic matrix where each entry $a_{\ell k}$ is given by

$$a_{\ell k} = \begin{cases} \frac{1}{n_k}, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.84}$$

2. **Laplacian rule:** generates a symmetric and doubly-stochastic matrix with the entries $\{a_{\ell k}\}$ given by

$$a_{\ell k} = \begin{cases} \frac{1}{n_{\max}}, & \ell \neq k, \ \ell \in \mathcal{N}_k, \\ 1 - \frac{n_k - 1}{n_{\max}}, & \ell = k, \\ 0, & \text{otherwise} \end{cases} \tag{2.85}$$

   where $n_{\max}$ is the maximum agent degree over the network.

3. **Maximum degree rule:** generates a symmetric and doubly-stochastic matrix with the entries $\{a_{\ell k}\}$ given by

$$a_{\ell k} = \begin{cases} \frac{1}{N}, & \ell \neq k, \ \ell \in \mathcal{N}_k, \\ 1 - \frac{n_k - 1}{N}, & \ell = k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.86}$$

4. **Metropolis rule:** generates a symmetric and doubly-stochastic matrix with the entries $\{a_{\ell k}\}$ given by

$$a_{\ell k} = \begin{cases} \frac{1}{\max(n_k, n_\ell)}, & \ell \neq k,\ \ell \in \mathcal{N}_k, \\ 1 - \sum_{m \in \mathcal{N}_k^-} a_{mk}, & \ell = k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.87}$$

5. **Relative degree:** gives more weight to agents that are better connected. This rule generates a left-stochastic matrix with the entries $\{a_{\ell k}\}$ given by

$$a_{\ell k} = \begin{cases} \frac{n_\ell}{\sum_{m \in \mathcal{N}_k} n_m}, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.88}$$

6. **Relative degree-variance:** assumes that the noise variances of the agents are known, or can be estimated from the data. The relative degree-variance rule generates a left-stochastic matrix where each entry $a_{\ell k}$ is given by

$$a_{\ell k} = \begin{cases} \frac{n_\ell\, \sigma_{v,\ell}^2}{\sum_{m \in \mathcal{N}_k} n_m\, \sigma_{v,m}^2}, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.89}$$

There are other methods to design the combination weights adaptively and optimally [34, 35]. In this thesis we focus only on the static design rules. However, the topology itself, the number of neighbors, and the distribution of the agents affect the performance of the network [36].

2.5.1.3 Network Objective

We associate a twice-differentiable individual cost function $J_k(w) \in \mathbb{R}$ with each agent. The objective of the network is to seek the unique minimizer of the aggregate cost function $J^{\text{glob}}(w)$ defined by

$$J^{\text{glob}}(w) \triangleq \sum_{k=1}^{N} J_k(w). \tag{2.90}$$

It is assumed that the individual cost functions $\{J_k(w)\}$ are each twice-differentiable and convex with at least one of them being ν_d–strongly convex. The aggregate cost function $J^{\text{glob}}(w)$ is also twice-differentiable and satisfies

$$0 < \nu_d I_M \leq \nabla_w^2 J^{\text{glob}}(w) \leq \delta_d I_M \tag{2.91}$$

for $\nu_d \leq \delta_d$. Under these conditions, the cost $J^{\text{glob}}(w)$ has a unique minimizer denoted by w°. It is not required that all individual costs $J_k(w)$ should be strongly convex. It is sufficient to assume that at least one of these costs is ν_d–strongly convex while the remaining costs are simply convex. This condition ensures that $J^{\text{glob}}(w)$ will be strongly convex as well.

There are several distributed strategies that can be used to seek the minimizer of (2.90), i.e.,

$$w^\circ = \arg\min_{w} \sum_{k=1}^{N} J_k(w). \tag{2.92}$$

We consider three distributed strategies, namely, the consensus and the two diffusion strategy ATC and CTA.

2.5.2 Consensus Strategy

In this strategy, at each time instant i every agent k performs the following two steps:

$$\boldsymbol{\psi}_{k,i-1} = \sum_{\ell=1}^{N} a_{\ell k} \boldsymbol{w}_{\ell,i-1} \tag{2.93}$$

$$\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \widehat{\nabla_{w^\intercal} J_k}(\boldsymbol{w}_{k,i-1}). \tag{2.94}$$

Each agent aggregates the iterates from its neighbors and updates this aggregate value by the gradient vector evaluated at its existing iterate $\boldsymbol{w}_{k,i-1}$. The intermediate iterate that results from the neighborhood combination is denoted by $\boldsymbol{\psi}_{k,i-1}$.

Consensus LMS:
For the mean-square-error network example the consensus strategy is given by the following form:

$$\boldsymbol{\psi}_{k,i-1} = \sum_{\ell=1}^{N} a_{\ell k} \boldsymbol{w}_{\ell,i-1} \tag{2.95}$$

$$\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} + 2\mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{w}_{k,i-1}). \tag{2.96}$$

2.5.3 Diffusion Strategies

The consensus strategy consists of two asymmetrically steps, namely, the decentralization step (2.94) and the cooperation term (2.93), which involves a convex combination.

The asymmetry in the consensus update is problematic when the strategy is used for adaptation and learning over networks, where it can cause an unstable growth of the state of the network (see [22, pp. 568,569]). However, diffusion strategies solve the asymmetry problem.

There are several types of distributed diffusion strategies [11, 12, 16, 37–39]. Two popular ones strategies are called Adapt-then-Combine (ATC) and Combine-then-Adapt (CTA) [40, 41].

2.5.3.1 Adapt-Then-Combine Diffusion Strategy (ATC)

In the ATC diffusion strategy, the first operation is the adaptation step where agent k uses its approximate gradient vector to update $\boldsymbol{w}_{k,i-1}$ to the intermediate estimate $\boldsymbol{\psi}_{k,i}$. The agents perform the adaptation step simultaneously to update their iterates $\boldsymbol{w}_{k,i-1}$ to the intermediate iterates $\boldsymbol{\psi}_{k,i}$ by using information from their neighbors. The second step is an aggregation step where agent k combines the intermediate iterates $\{\boldsymbol{\psi}_{\ell,i}\}$ from its neighbors to obtain its updated iterate $\boldsymbol{w}_{k,i}$.

To summarize, at each time instant i every agent k performs the following two steps:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu_k \widehat{\nabla_{w^\intercal} J_k}(\boldsymbol{w}_{k,i-1}) \tag{2.97}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} a_{\ell k} \boldsymbol{\psi}_{\ell,i}. \tag{2.98}$$

2.5.3.2 Combine-Then-Adapt Diffusion Strategy (CTA)

On contrast of ATC, the first operation in the CTA diffusion strategy is an aggregation step where each agent k combines the existing iterates from its neighbors $\{\boldsymbol{w}_{\ell,i-1}\}$ to obtain the intermediate iterate $\boldsymbol{\psi}_{k,i-1}$. The second operation is an adaptation step where each agent k approximates its gradient vector and uses it to update its intermediate iterate to $\boldsymbol{w}_{k,i}$. In summary, at each time instant i every agent k performs the following two steps:

$$\boldsymbol{\psi}_{k,i-1} = \sum_{\ell=1}^{N} a_{\ell k} \boldsymbol{w}_{\ell,i-1} \tag{2.99}$$

$$\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \widehat{\nabla_{w^\intercal} J_k}(\boldsymbol{\psi}_{k,i-1}). \tag{2.100}$$

Note that the gradient vector in the diffusion strategies is now evaluated at $\boldsymbol{\psi}_{k,i-1}$ [40]. In addition, other forms of diffusion strategies are possible by enlarging the cooperation among the agents through, for instance exchanging the gradient vector approximations (see Appendix A.1).

2.5.3.3 Diffusion LMS Networks

For the mean-square-error network example, the ATC diffusion strategy is given by the following form:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + 2\mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}}(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{w}_{k,i-1}) \tag{2.101}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} a_{\ell k}\boldsymbol{\psi}_{\ell,i}. \tag{2.102}$$

In contrast, the CTA diffusion strategy is given by

$$\boldsymbol{\psi}_{k,i-1} = \sum_{\ell=1}^{N} a_{\ell k}\boldsymbol{w}_{\ell,i-1} \tag{2.103}$$

$$\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} + 2\mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}}(\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i}\boldsymbol{\psi}_{k,i-1}). \tag{2.104}$$

2.5.4 Stability of First, Second, and Fourth-Order Error Moments

We describe the non-cooperative, consensus, and diffusion strategies by means of a single description as follows:

$$\boldsymbol{\phi}_{k,i-1} = \sum_{\ell=1}^{N} a_{1,\ell k}\boldsymbol{w}_{\ell,i-1} \tag{2.105}$$

$$\boldsymbol{\psi}_{k,i} = \sum_{\ell=1}^{N} a_{0,\ell k}\boldsymbol{\phi}_{\ell,i-1} - \mu_k \widehat{\nabla_{w^{\mathsf{T}}} J_k}(\boldsymbol{\phi}_{k,i-1}) \tag{2.106}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} a_{2,\ell k}\boldsymbol{\psi}_{\ell,i} \tag{2.107}$$

where the $N \times N$ matrices A_0, A_1, and A_2 are left-stochastic and satisfy

$$A_0^{\mathsf{T}}\mathbb{1} = \mathbb{1},\ A_1^{\mathsf{T}}\mathbb{1} = \mathbb{1},\ A_2^{\mathsf{T}}\mathbb{1} = \mathbb{1}. \tag{2.108}$$

Furthermore, the left-stochastic matrix P is defined by

$$P \triangleq A_1 A_0 A_2. \tag{2.109}$$

We assume that P is a primitive matrix (see Appendix A.2). Now, different choices of $\{A_0, A_1, A_2\}$ correspond to different distributed strategies as follows:

$$\text{non-cooperative:} \quad A_1 = A_0 = A_2 = I \quad \rightarrow P = I \tag{2.110}$$

$$\text{consensus:} \quad A_0 = A,\ A_1 = A_2 = I \rightarrow P = A \tag{2.111}$$

$$\text{ATC diffusion:} \quad A_2 = A,\ A_0 = A_1 = I \rightarrow P = A \tag{2.112}$$

$$\text{CTA diffusion:} \quad A_1 = A,\ A_0 = A_2 = I \rightarrow P = A. \tag{2.113}$$

In addition, the aggregate cost in (2.90) and the individual costs $J_k(w)$ satisfy conditions (2.11) and (2.12) as well as the smoothness condition (see [22, p. 546]) are assumed to hold. Moreover, we assume that the first and second-order moments of the gradient noise process satisfy the conditions in [22, pp. 496, 497]. Then, for sufficiently small step-sizes it holds that

$$\limsup_{i\rightarrow\infty} \ \|\mathbb{E}\ \widetilde{\boldsymbol{w}}_{k,i}\| = \mathcal{O}(\mu_{\max}) \tag{2.114}$$

$$\limsup_{i\rightarrow\infty} \ \mathbb{E}\ \|\widetilde{\boldsymbol{w}}_{k,i}\|^2 = \mathcal{O}(\mu_{\max}) \tag{2.115}$$

$$\limsup_{i\rightarrow\infty} \ \mathbb{E}\ \|\widetilde{\boldsymbol{w}}_{k,i}\|^4 = \mathcal{O}(\mu^2_{\max}) \tag{2.116}$$

where $\mu_{\max}$ is the maximum step-size over the network and $k = 1, 2, \ldots, N$.

2.5.5 Simulations

We present a simulation example of an MSE network. The network consists of $N = 30$ agents. Figure 2.9 depicts the statistical profile showing how the signal and noise power vary across the agents. The regressors are of size $M = 2$, zero-mean Gaussian, independent in time and space and have covariance matrices $\{R_{u,k}\}$. The noise variance of each agent k is denoted by $\sigma^2_{v,k}$. Figure 2.10 shows the learning curves for different diffusion LMS algorithms, the non-cooperation case, and the global (centralized) solution in terms of EMSE (ER) and MSD. We use $\mu = 0.05$ for all agents and the uniform rule to generate the diffusion matrix A. The results are averaged over 100 independent experiments.

2.6 Optimization in Mobile Networks

Mobility allows agents to move to more appropriate locations to improve the quality of their local observations and towards the desired location when the network is tracking a target. The objective of this section is to present and study mobile adaptive networks.

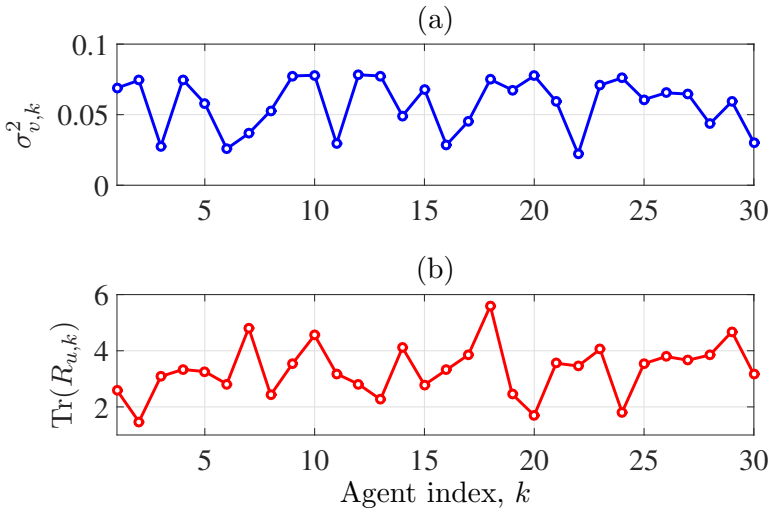

Figure 2.9. Statistical noise and signal profiles over the network.

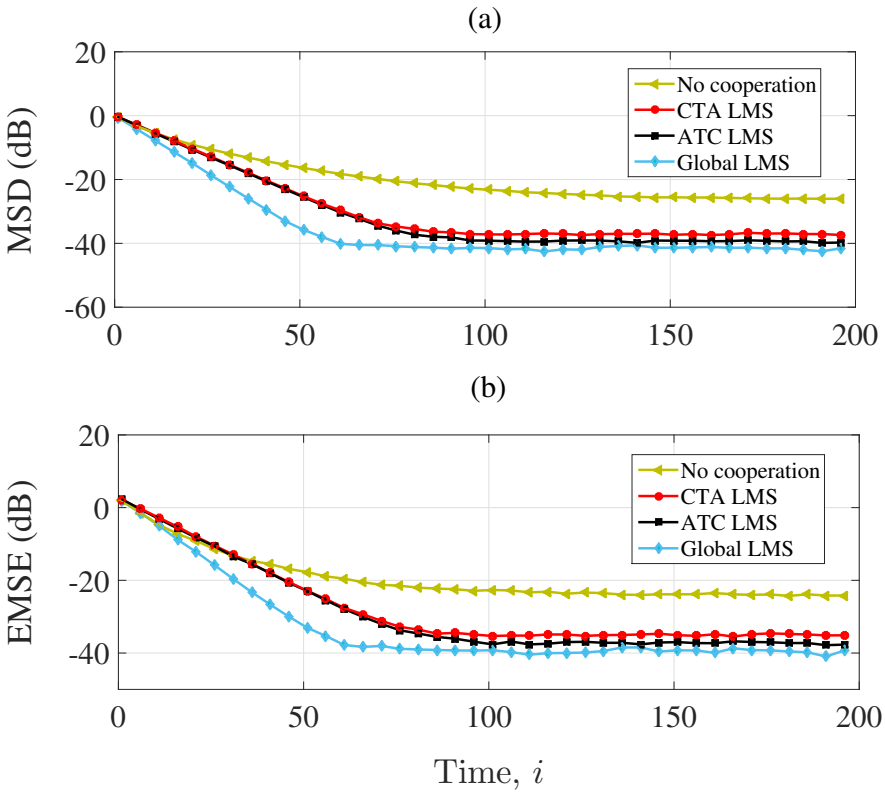

Figure 2.10. Learning curves MSD and EMSE of several strategies, namely, the non-cooperative case, ATC, CTA, and the global (centralized) solution.

Collective motion has been observed in several natural phenomena where animals move together in amazing synchrony. The individual agents in these groups tend to have similar speeds and move in parallel while keeping a safe distance from their neighbors to avoid collisions. Example for this are fish schools [42–46], honeybees swarming towards a hive [47], birds flying in V-formations [45, 46], and bacteria motility [48, 49]. The works in [50] mimic several collective behaviors of animals groups in fixed and dynamic topologies.

When applying adaptive diffusion techniques to guide the self-organization process, considering harmonious motion and collision avoidance is proposed in many works [42–46, 51]. The works in [42, 52] developed a diffusion algorithm for mobile adaptive networks that can move coherently towards a target of unknown location. In [43, 44] a diffusion adaptation model to simulate the behavior of fish schools in the presence of predators was proposed. The authors of [53] developed an algorithm for fish to form a school while foraging for food. The works in [54, 55] study adaptive networks where only a fraction of the agents are assumed to be informed, while the remaining agents are uninformed. Informed agents collect data and perform the network tasks, while the uninformed agents only follow and support the informed ones. Furthermore, in [56] a modified ATC diffusion algorithm for mobile adaptive networks is proposed where the individual agents are allowed to move in pursuit of a target.

In the following we present diffusion strategies for mobile networks that (i) possess distributed adaptation abilities and (ii) exhibits collective patterns of motion. The strategies involve two diffusion steps: one for estimating the location of a target and another for tracking the centeral velocity of the network. The strategies include velocity and location control mechanisms to control the motion of the agents in a distributed manner. We will show via some simulations how the algorithms mimic the coherent motion of fish schools and let the agents avoid the obstacles.

2.6.1 Network Model

The goal is to estimate a target location w° and let the agents move coherently towards this target. At each time instant i every agent k has access to a scalar measurement $\boldsymbol{d}_k(i)$ and a $1 \times M$ regression data vector $\boldsymbol{u}_{k,i}$. The measurements across all agents are assumed to be related via the linear regression model:

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^\circ + \boldsymbol{n}_k(i) \tag{2.117}$$

where $\boldsymbol{n}_k(i)$ is a noise process that is assumed to be zero-mean white and independent of all other variables; all other random process are assumed to be stationary. Figure 2.11

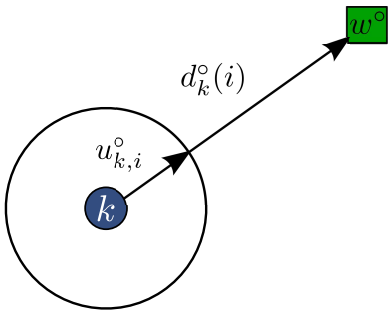

Figure 2.11. Distance and direction of the target w° from agent k at location $x_{k,i}$. The true unit direction vector $u_{k,i}^\circ$ points towards the target w°.

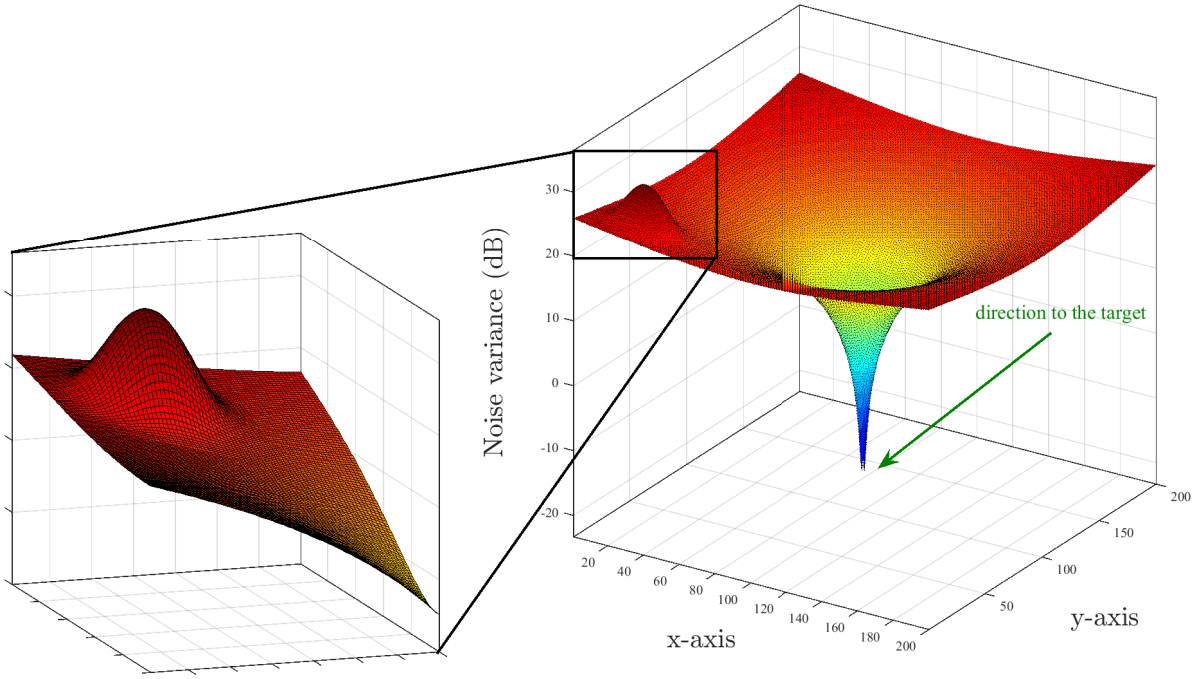

Figure 2.12. The noise variance over the region of interest.

shows the illustration of the data in the mobile environment. The true distance between the target located at w° and agent k located at $x_{k,i}$ at time instant i is given by

$$d_k^\circ(i) = u_{k,i}^\circ(w^\circ - x_{k,i}) \tag{2.118}$$

where $u_{k,i}^\circ$ denotes the true unit direction vector pointing from $x_{k,i}$ to w°. This vector is given by

$$u_{k,i}^\circ = \frac{(w^\circ - x_{k,i})^\mathsf{T}}{\|w^\circ - x_{k,i}\|}. \tag{2.119}$$

The gents observe noisy measurements of $u_{k,i}^\circ$ and $d_k^\circ(i)$. Starting from [42], the noisy location of the target is denoted by $\boldsymbol{q}_{k,i}$ and given by

$$\boldsymbol{q}_{k,i} = x_{k,i} + \boldsymbol{d}_k(i)\boldsymbol{u}_{k,i}^\mathsf{T} = w^\circ + \boldsymbol{\eta}_{k,i} \tag{2.120}$$

where $\boldsymbol{\eta}_{k,i}$ is assumed to be a zero-mean random process with covariance matrix

$$\Pi_k(i) = \rho\|w^\circ - x_{k,i}\|^2 I_M \tag{2.121}$$

for some small constant $\rho > 0$. Furthermore, the noise variance is given by $\sigma_k^2(i) = \mathrm{Tr}(\Pi_k(i))$. Figure 2.12 shows an example of the noise variance values around the target location. The closer the agent is to the target location, the lower is the influence of the noise level. Note that we include a region of high noise, which the agents will have to avoid and navigate around in the simulations.

2.6.2 Motion Mechanism

Each agent k updates its location vector according to the following relation:

$$x_{k,i+1} = x_{k,i} + \Delta t \cdot v_{k,i+1} \tag{2.122}$$

where Δt is a positive time step. $v_{k,i+1}$ is the velocity vector of agent k and is computed as

$$v_{k,i+1} = \lambda \cdot h(\boldsymbol{w}_{k,i} - x_{k,i}) + \alpha \cdot \frac{g_{k,i}}{\|g_{k,i}\|} + \beta \cdot \boldsymbol{v}_{k,i}^g + \gamma \cdot \delta_{k,i} \tag{2.123}$$

where $\{\lambda, \alpha, \beta, \gamma\}$ are non-negative weighting factors. Figure 2.13 shows some terms of the velocity equation. The first term of (2.123) allows the agent to move towards the target where $v_{k,i}^b = \boldsymbol{v}_{k,i}^g$. The function $h(\cdot)$ is defined in [42] as follows:

$$v_{k,i}^a = h(\boldsymbol{w}_{k,i} - x_{k,i}) = \begin{cases} \boldsymbol{w}_{k,i} - x_{k,i}, & \text{if } \|\boldsymbol{w}_{k,i} - x_{k,i}\| \leq s, \\ s \cdot \frac{\boldsymbol{w}_{k,i} - x_{k,i}}{\|\boldsymbol{w}_{k,i} - x_{k,i}\|}, & \text{otherwise} \end{cases} \tag{2.124}$$

where s is some positive scaling factor used to bound the speed in pursuing the target. $\boldsymbol{w}_{k,i}$ is the local estimate of the target's location. The second term relates to moving

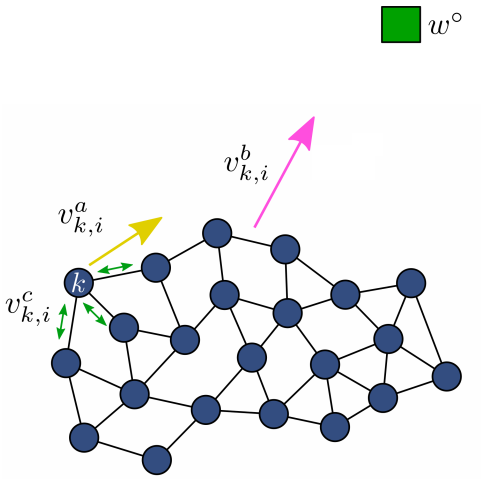

Figure 2.13. Illustration of the main terms of the velocity equation.

towards regions with lower noise level to improve the estimation performance. The vector $g_{k,i}$ is given by

$$g_{k,i} = -\sum_{\ell \in \mathcal{N}_{k,i}^-} \left(\sigma_\ell^2(i) - \sigma_k^2(i)\right) \frac{x_{\ell,i} - x_{k,i}}{\|x_{\ell,i} - x_{k,i}\|} \tag{2.125}$$

where $\mathcal{N}_{k,i}^-$ is the set of neighbors of the agent k, excluding k. Note that the network topology is not fixed and changes over time. The term related to the vector $\boldsymbol{v}_{k,i}^g$, which is the local estimate of the central velocity of the network, is needed for allowing the agents to adjust their velocities to be consistent with the average displacement vector in the neighborhood in harmonic motion. Finally, the last term allows the agents to maintain a safe distance r between each other to avoid collisions. The vector $\delta_{k,i}$ is given by

$$v_{k,i}^c = \delta_{k,i} = \sum_{\ell \in \mathcal{N}_{k,i}^-} \left(\|x_{\ell,i} - x_{k,i}\| - r\right) \frac{x_{\ell,i} - x_{k,i}}{\|x_{\ell,i} - x_{k,i}\|}. \tag{2.126}$$

The vectors $\boldsymbol{v}_{k,i}^g$ and $\boldsymbol{w}_{k,i}$ are estimated using the diffusion strategy as will be explained in the next chapters.

2.6.3 Mean-Square-Errors

We are interested in three error quantities when considering mobile networks:

1. The network mean-square deviation MSD for estimating the target location, which is given by

$$\text{MSD}_w(i) \triangleq \frac{1}{N} \sum_{k=1}^{N} \mathbb{E}\, \|\widetilde{\boldsymbol{w}}_{k,i}\|^2 \tag{2.127}$$

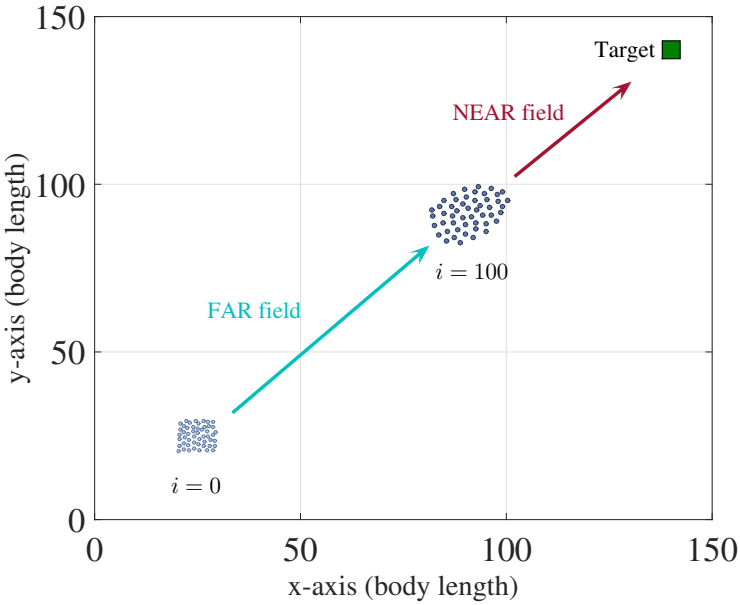

Figure 2.14. Tracking behavior of a mobile network in the far and near fields.

where

$$\widetilde{\boldsymbol{w}}_{k,i} = w^{\circ} - \boldsymbol{w}_{k,i}. \tag{2.128}$$

2. The network mean-square-error for estimating the velocity of the network mass, which is given by

$$\mathrm{MSE}_v(i) \triangleq \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\, \|\widetilde{\boldsymbol{v}}^{g}_{k,i}\|^2 \tag{2.129}$$

where

$$\widetilde{\boldsymbol{v}}^{g}_{k,i} = v^{\circ}_{i} - \boldsymbol{v}^{g}_{k,i}. \tag{2.130}$$

The average velocities of all agents in the network is denoted by v°_{i}.

3. The network mean-square disagreement on velocity (the velocity of agent k relative to the velocity of the network mass), which is given by

$$\mathrm{D}_v(i) \triangleq \frac{1}{N}\sum_{k=1}^{N} \mathbb{E}\, \|\widetilde{\boldsymbol{v}}_{k,i}\|^2 \tag{2.131}$$

where

$$\widetilde{\boldsymbol{v}}_{k,i} = v^{\circ}_{i} - \boldsymbol{v}_{k,i}. \tag{2.132}$$

The performance of motion and velocity are measured in the far field, while the steady-state performance of the target location estimation is measured in the near field, see Fig. 2.14.

2.6.4 Simulations

Fish schooling:
We simulate the motion of mobile networks with 50 agents using the method and the parameters proposed in [42].

Figure 2.15 illustrates the maneuver of a mobile network over time. The unit length is the body length of an agent. The green square denotes the target of interest. We place a region with high noise variance along the way to the target as shown in Fig. 2.12. When the network approaches the region with high noise, the swarm splits and navigates around it. Afterwards, the agents regroup again to continue the schooling process. Finally, the network successfully arrives at the target.

Figures 2.16, 2.17, and 2.18 show the network mean-square-errors which are evaluated using different values of the parameter λ for the ATC diffusion algorithm and the non-cooperative case.

2.7 Optimization in Multi-Task Networks

In the previous sections, the agents seek a common objective to find the global minimizer for some cost function. However, there are important situations where different agents in a network are interested in different objectives [57–60]. These situations arise frequently in clustering problems where a subset of the agents belongs to one cluster and another subset belongs to a second cluster. Cooperation among agents with different objectives leads to undesired results. Therefore, the agents need to figure out which subset of their neighbors share their objective.

In this section, we present multi-task networks. Single-task networks are those where all agents are interested in the same objective and sensing data is generated by only one source, while the agents in multi-task networks sense data generated by different sources and they are, therefore, interested in different objectives. We explain later how to design the combination weights such that the agents are able to cluster and cooperate only with neighbors that share the same objective.

2.7.1 Network Model

Consider a network with N agents connected by a graph. In addition, assume that there are C clusters, denoted by $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_C$, where each $\mathcal{C}_m$ represents the set of

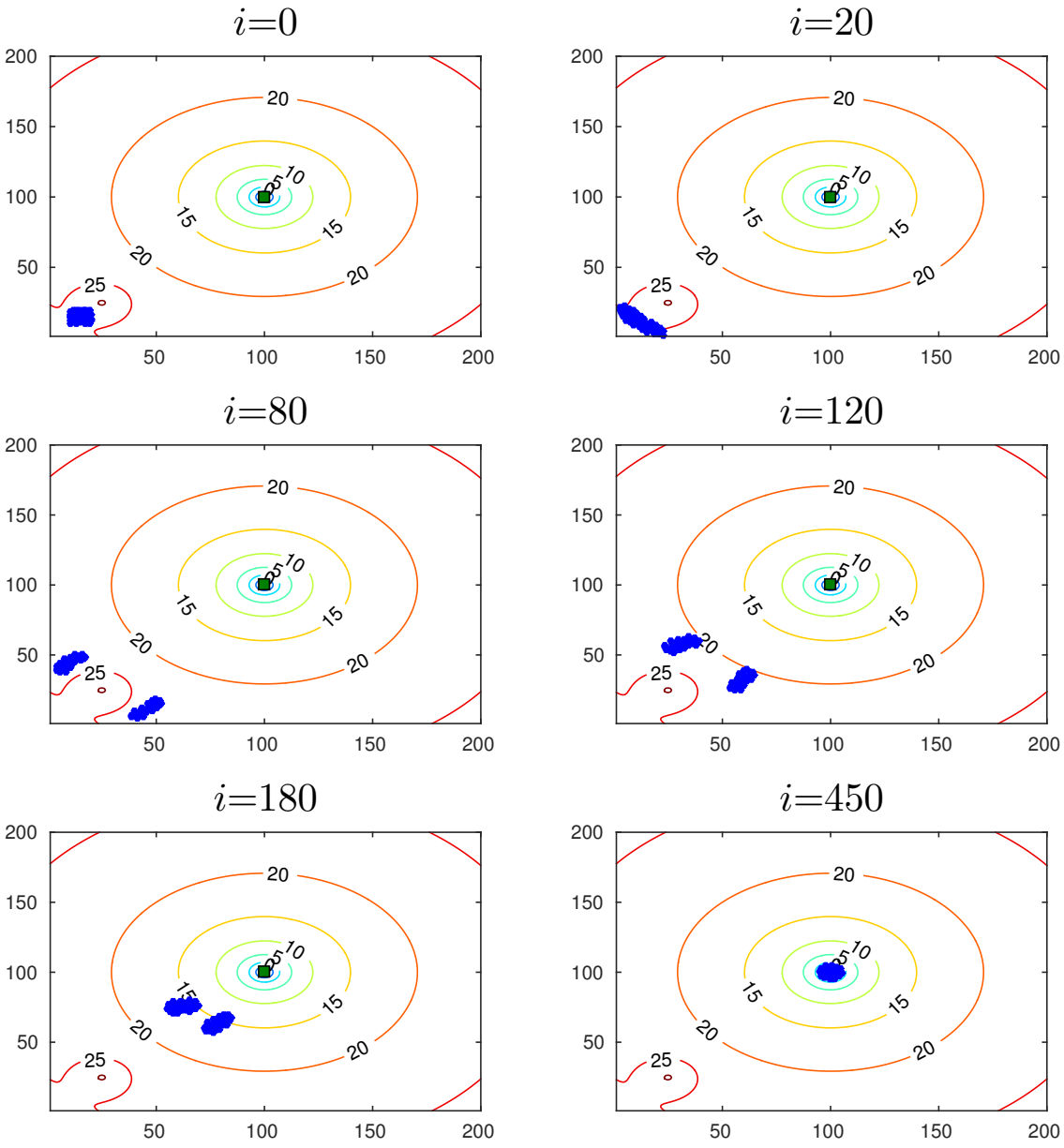

Figure 2.15. Maneuver of a fish school moving towards one food source over time. The length unit of the x- and y-axis is the body length of the agents.

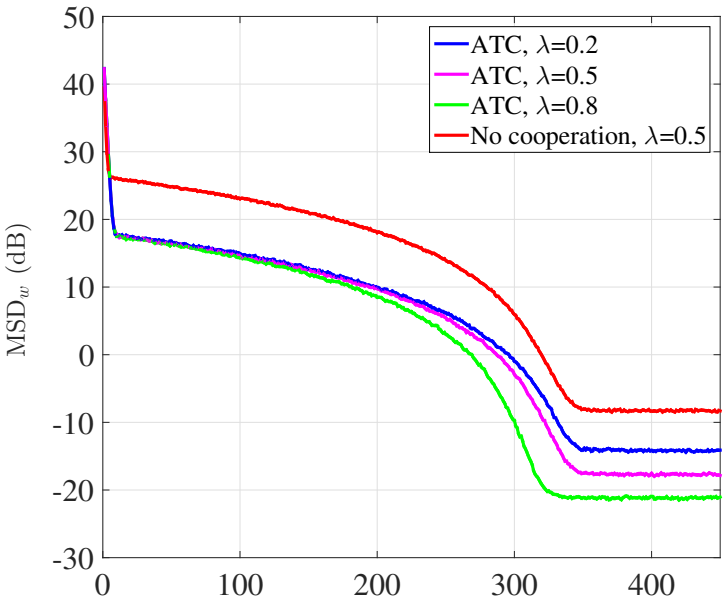

Figure 2.16. Transient network mean-square deviation for estimating the target location, w°.

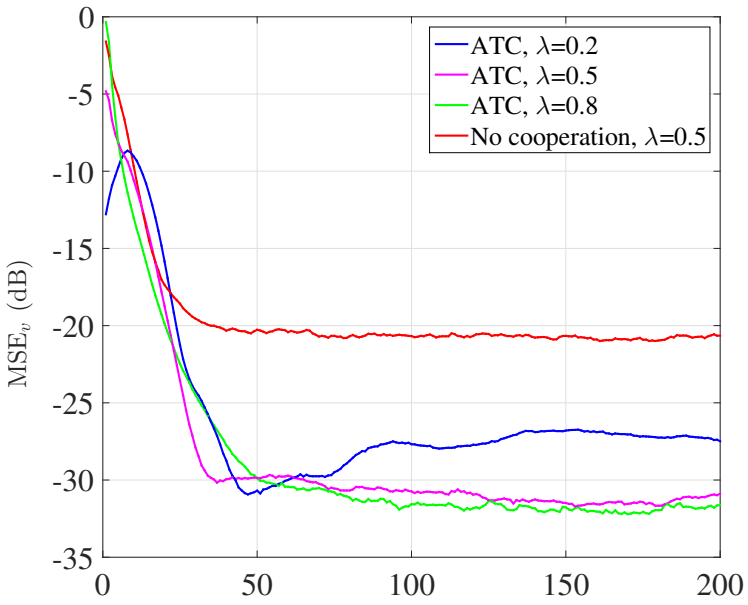

Figure 2.17. Transient network mean-square performance for estimating the central velocity v° in the far field.

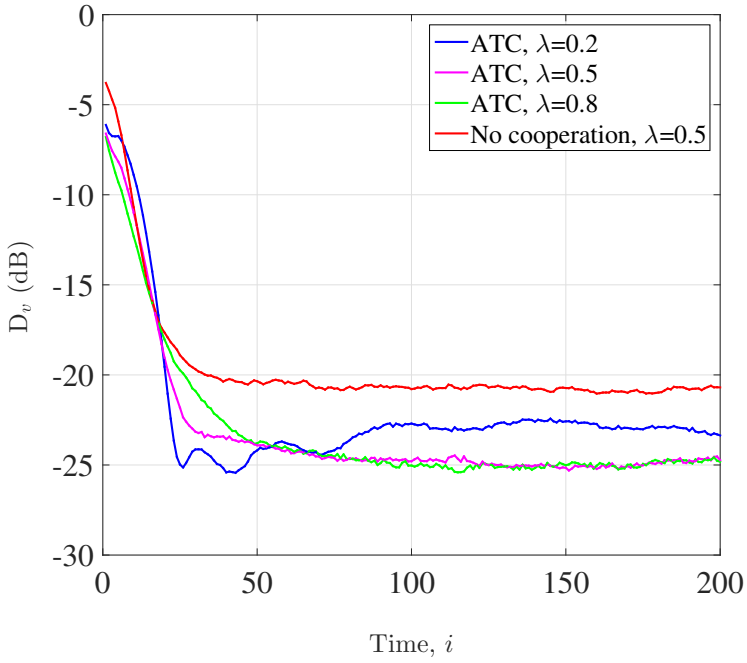

Figure 2.18. Transient network mean-square disagreement of the velocities in the far field.

agent indices in that cluster, $m \in \{1, 2, \ldots, C\}$. We associate an unknown column vector of size $M \times 1$ with each cluster, denoted by $w^{\circ}_{\mathcal{C}_m} \in \mathbb{R}^M$.

Each agent k wishes to recover the model for its cluster; the unknown model for agent k is denoted by $\{w^{\circ}_k\}$. Figure 2.19(a) presents a network with only one task (objective), while Fig. 2.19(b) shows a multi-task network with three clusters represented by three different colors. All agents in the same cluster are interested in estimating the same parameter vector. The cluster information is not known to the agents beforehand.

2.7.2 Cost Function

We associate with each agent k a strongly-convex and twice-differentiable individual cost function, denoted by $J_k(w) \in \mathbb{R}$ with a unique minimum at w°_k. The objective of the network is to seek the minimizer of the aggregate cost function $J^{\text{glob}}(w)$ over the vectors $\{w_k\}$. The aggregate cost function is defined by

$$J^{\text{glob}}(w_1, w_2, \ldots, w_N) \triangleq \sum_{k=1}^{N} J_k(w_k). \tag{2.133}$$

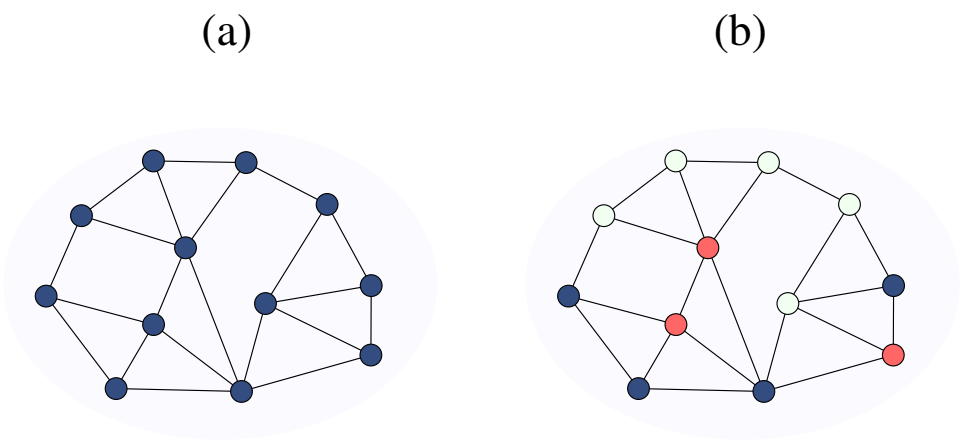

Figure 2.19. Example of a distributed single-task network (a) and a distributed multi-task network (b).

Since agents from different clusters do not share the same minimizers, the aggregate cost can be rewritten as follows:

$$J^{\text{glob}}(w_{\mathcal{C}_1}, \ldots, w_{\mathcal{C}_C}) \triangleq \sum_{m=1}^{C} \sum_{k \in \mathcal{C}_m} J_k(w_k) \tag{2.134}$$

where the model of agent k agrees with the model of the cluster that k belongs to, i.e., $w_k^\circ = w_{\mathcal{C}_m}^\circ$ if $k \in \mathcal{C}_m$.

2.7.3 State-of-the-Art in Multi-Task Networks

There have been several useful works in the literature on the solution of inference problems for multi-task networks, i.e., for networks with multiple unknown models (tasks) — see, e.g., [30, 33, 58, 61–67] and the references therein.

In the solutions developed in [64–66], clustering is achieved by relying on adaptive combination strategies, whereby weights on edges between agents are adapted and their size becomes smaller for unrelated tasks. In these earlier works, there still exists the possibility that valid links between agents belonging to the same cluster may be overlooked, mainly due to errors during the adaptation process. The work in [33] assumed that all agents know a priori the relationship between their tasks and the tasks of their neighbors, where some of these tasks are local and some are global for the whole agents. The related version of this work in [58] instead assumes node-specific parameters. The approach in this work is motivated by results from [67] where the clustering and learning operations were decoupled from each other. In this way, tracking errors do not influence the clustering mechanism and the resulting distributed algorithm enables the

agents to identify their clusters and to attain improved learning accuracy. The work in [67] evaluated the error probabilities of types I and II, i.e., of false alarm and misdetection for their proposed scheme and showed that these errors decay exponentially with the step-size. This means that the probability of correct clustering can be made arbitrarily close to one by selecting sufficiently small step-sizes. Still, it is preferable to *merge* the clustering and learning mechanisms rather than have them run separately of each other. Doing so reduces the computational burden and, if successful, can also lead to an enhancement in clustering accuracy relative to the earlier approaches [64–66].

In contrast, the works in [63, 68, 69] employ diffusion strategies to develop distributed algorithms that address clustered multi-task problems by minimizing an appropriate mean-square-error criterion with the regularization concept. In the same context, paper [13] introduces grouping into diffusion adaptation to take advantage of structural similarities between parameter vectors that are estimated over the multi-task network.

For some multi-task networks, the agents need to reach an agreement on only one common objective. In the earlier works [62, 70], the agents are subject to data arising from two different models and the aim of the network is to reach an agreement to track only one of these two observed sources in a distributed manner. The scheme of the algorithm proposed in [62] is based on binary labeling. This implies that the algorithm can be applied for *only* two different observed models.

Moreover, applying [62] in a mobile environment leads to some undesired scenarios, e.g., splitting the network into two sub-networks before reaching an agreement. The classification scheme proposed in [62], which determines the subset of neighbors that observes the same model, has some disadvantages. First, the performance of this scheme depends on the initial location of the network and the location of the models. Second, since the decision-making objective depends on the classification output, errors made in the classification process have an impact on the global decision. Therefore, a fast clustering technique that lets the agents distinguish between the neighbors in real-time is needed in mobile networks because the topology changes quickly due to the movement of the agents. Third, changes in the topology over time imply that the network may be separated into two groups before reaching the agreement on one model. Groups moving far away from each other towards their different desired models would lose the connections between each other. This means that the decision-making process will have to fail to ensure that the network converges to only one desired model. In the presence of multiple targets, another situation is considered in [71] where the agents switch the target they are tracking and form distinct clusters. The resulting clusters split up while moving and pursue their distinct target over time.

To proceed with multi-task networks, the first step is to provide an effective, accurate, and fast decentralized clustering algorithm aimed at identifying and forming clusters of agents of similar objectives, and at guiding cooperation to enhance the inference performance in real-time. In the sequel, we propose decentralized clustering and linking algorithms by networked agents [17, 18].

Chapter 3

Decentralized Clustering and Linking by Networked Agents

'If you can't explain it simply, you don't understand it well enough.'

Albert Einstein

We consider the problem of decentralized clustering and estimation over multi-task networks, where agents infer and track different models of interest. The agents do not know beforehand which model is generating their own data. They also do not know which agents in their neighborhood belong to the same cluster. We propose a decentralized clustering algorithm aimed at identifying and forming clusters of agents of similar objectives, and at guiding cooperation to enhance the inference performance. One key feature of the proposed technique is the integration of the learning and clustering tasks into a single strategy. We analyze the performance of the procedure and show that the error probabilities of types I and II decay exponentially to zero. While links between agents following different objectives are ignored in the clustering process, we nevertheless show how to exploit these links to relay critical information across the network for enhanced performance. Simulation results illustrate the performance of the proposed method in comparison to other state-of-the-art techniques[1].

3.1 Introduction

Distributed learning is a powerful technique for extracting information from networked agents (see, e.g., [22, 26, 27, 29–31] and the references therein). In this chapter, we consider a network of agents connected by a graph. Each agent senses data generated by some *unknown* model. It is assumed that there are clusters of agents within the network, where agents in the same cluster observe data arising from the same model.

However, the agents do not know which model is generating their own data. They also do not know which agents in their neighborhood belong to the same cluster. Scenarios

[1]This chapter is based on the journal article: S. Khawatmi, A. H. Sayed, and A. M. Zoubir, "Decentralized clustering and linking by networked agents," *IEEE Trans. Signal Processing*, vol. 65, pp. 3526–3537, July 2017.

of this type arise, for example, in tracking applications when a collection of networked agents is tasked with tracking several moving objects [71–73]. Clusters end up being formed within the network with different clusters following different targets. The quality of the tracking/estimation performance will be improved if neighboring agents following the same target know of each other to promote cooperation. It is not only cooperation within clusters that is useful, but also cooperation across clusters, especially when targets move in formation and the location of the targets are correlated. Motivated by these considerations, the main objective of this work is to develop a distributed technique that enables agents to recognize neighbors from the same cluster and promotes cooperation for improved inference performance.

A useful strategy for clustering over adaptive networks was proposed in [64], relying on the use of adaptive combination weights. This strategy was further refined in [65] to reduce its sensitivity to initial conditions. In order to avoid this difficulty and obtain a more robust method, the authors in [67] proposed an alternative construction where the clustering and inference tasks are separated from each other. For sufficiently small step-sizes, this approach was shown to lead to probabilities of error that decay exponentially to zero. Motivated by [67], we propose a modified strategy where we merge the clustering and inference tasks, thus reducing the computation burden while enhancing the accuracy of the clustering step compared to [64, 65].

The solutions developed in [64–66], clustering is achieved by relying on adaptive combination strategies. In these works, there still exists the possibility that valid links between agents belonging to the same cluster may be overlooked, mainly due to errors during the adaptation process. The clustering and learning operations in [67] were decoupled from each other. In this way, tracking errors do not influence the clustering mechanism and the resulting distributed algorithm enables the agents to identify their clusters and to attain improved learning accuracy.

In this chapter, we *merge* the clustering and learning mechanisms rather than have them run separately of each other. Doing so reduces the computational burden and, if successful, can also lead to enhancement in clustering accuracy compared to the earlier approaches [64–66].

We showed in [18] that this is indeed possible for a particular class of inference problems involving mean-square-error risks. In this chapter, we generalize the results and devise an *integrated* clustering-learning approach for general-purpose risk functions. Additionally, and motivated by the results from [62] on adaptive decision-making by networked agents, we further incorporate a smoothing mechanism into our strategy to enhance the belief that agents have about their clusters. We also show how to exploit the unused

links among neighboring agents belonging to different clusters to *relay* useful information among agents. We carry out a detailed analysis of the resulting framework, and illustrate its superior performance by means of computer simulations.

The organization of this chapter is as follows. The network and data model are described in Sec. 3.2, while the integrated clustering and inference framework is developed in Sec. 3.3. The network error recursions are derived in Sec. 3.4, and the probabilities of erroneous decision are derived in Sec. 3.5. In Secs. 3.6 and 3.7 we illustrate two additional techniques for linking agents and labeling models, respectively. Finally, we present some simulation results in Sec. 3.8.

3.2 Network and Data Model

3.2.1 Network Overview

We consider a network with N agents connected by a graph. It is assumed that there are C clusters, denoted by $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_C$, where each $\mathcal{C}_m$ represents the set of agent indices in that cluster. We associate an unknown column vector of size $M \times 1$ with each cluster, denoted by $w^\circ_{\mathcal{C}_m} \in \mathbb{R}^M$. The aggregation of all these unknowns is denoted by

$$w^\circ_{\mathcal{C}} \triangleq \mathrm{col}\{w^\circ_{\mathcal{C}_1}, w^\circ_{\mathcal{C}_2}, \ldots, w^\circ_{\mathcal{C}_C}\},\ (CM \times 1). \tag{3.1}$$

Each agent k wishes to recover the model for its cluster; the unknown model for agent k is denoted by $\{w^\circ_k\}$. Obviously, this model agrees with the model of the cluster that k belongs to, i.e., $w^\circ_k = w^\circ_{\mathcal{C}_m}$ if $k \in \mathcal{C}_m$. We stack all $\{w^\circ_k\}$ into a column vector:

$$w^\circ \triangleq \mathrm{col}\{w^\circ_1, w^\circ_2, \ldots, w^\circ_N\},\ (NM \times 1). \tag{3.2}$$

Figure 3.1 illustrates a network with $C = 3$ clusters represented by three colors. All agents in the same cluster are interested in estimating the same parameter vector. We denote the set of neighbors of an agent k by $\mathcal{N}_k$. Observe in this example that the neighbors of agent k belong to different clusters. The cluster information is not known to the agents beforehand. For instance, agent k would not know that its neighbors are sensing data arising from different models. If we allow the network to perform indiscriminate cooperation, then performance will degrade significantly. For this reason, a clustering operation is needed to allow the agents to learn which neighbors to cooperate with towards the same objective. The technique developed in this work will allow

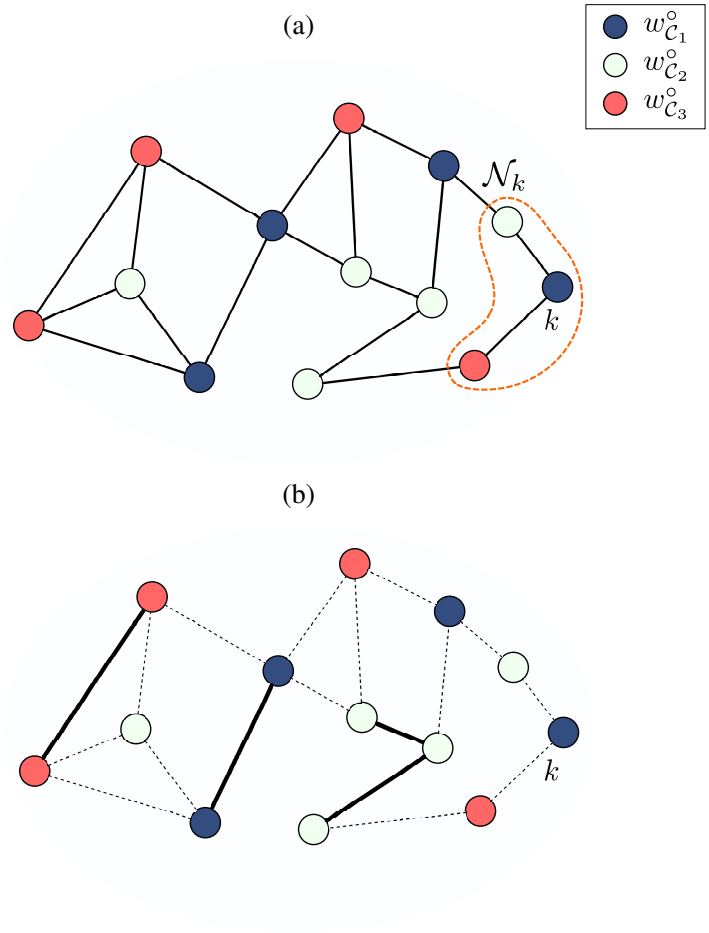

Figure 3.1. (*Top*) Example of a network topology involving three clusters, represented by three different colors. (*Bottom*) Clustered topology that will result for the network shown on top.

agents to emphasize links to neighbors in the same cluster and to disregard links to neighbors from other clusters. The outcome would be a graph structure similar to the one shown in the bottom part of the same figure, where unwarranted links are represented by dotted lines. In this way, the interference caused by different objectives is avoided and the overall performance for each cluster will be improved. Turning off a link between two agents means that there is no more sharing of data between them. Still, we will exploit these "unused" links by assigning to them a useful role in relaying information across the network.

3.2.2 Topology Matrices

In preparation for the description of the proposed strategy, we introduce the $N \times N$ adjacency matrix $Y = [y_{\ell k}]$, whose elements are either zero or one depending on whether

agents are linked by an edge or not. Specifically,

$$y_{\ell k} = \begin{cases} 1, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

We assume that each agent k belongs to its neighborhood set, $k \in \mathcal{N}_k$. The set $\mathcal{N}_k^-$ excludes k. Agents know their neighborhoods but they do not know which subset of their neighbors is subjected to data from the same model. In order to devise a procedure that allows agents to arrive at this information, we introduce a second $N \times N$ clustering matrix, denoted by $\boldsymbol{E}_i$ at time instant i, in a manner similar to the adjacency matrix Y, except that the value at location (ℓ, k) will be set to one if agent k *believes* at time instant i that its neighbor ℓ belongs to the same cluster:

$$\boldsymbol{e}_{\ell k}(i) = \begin{cases} 1, & \text{if } \ell \in \mathcal{N}_k \text{ and } k \text{ believes that } w_k^\circ = w_\ell^\circ, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

The entries of $\boldsymbol{E}_i$ will be learned online. At every time instant i, we can then use these entries to infer which neighbors of k are believed to belong to the same cluster as k; these would be the indices of the nonzero entries in the $k-$th column of $\boldsymbol{E}_i$. We collect these indices into the neighborhood set, $\boldsymbol{\mathcal{N}}_{k,i}$; this set is a subset of $\mathcal{N}_k$ and it evolves over time during the learning process. At any time instant i, agent k will only be cooperating with the neighbors within $\boldsymbol{\mathcal{N}}_{k,i}$. We will describe in the sequel how $\boldsymbol{E}_i$ is learned.

3.2.3 Problem Formulation

We associate with each agent k a strongly-convex and differentiable risk function $J_k(w_k)$, with a unique minimum at w_k°. In general, each risk $J_k(w_k) : \mathbb{R}^M \to \mathbb{R}$, is defined as the expectation of some loss function $Q_k(\cdot)$, say,

$$J_k(w_k) \;=\; \mathbb{E}\, Q_k(w_k; \boldsymbol{z}_k) \tag{3.5}$$

where $\boldsymbol{z}_k$ denotes random data sensed by agent k and the expectation is over the distribution of this data. The network of agents is interested in estimating the minimizers of the following aggregate cost over the vectors $\{w_k\}$:

$$J^{\text{glob}}(w_1, w_2, \ldots, w_N) \triangleq \sum_{k=1}^{N} J_k(w_k). \tag{3.6}$$

Since agents from different clusters do not share the same minimizers, the aggregate cost can be rewritten as

$$J^{\text{glob}}(w_{\mathcal{C}_1}, \ldots, w_{\mathcal{C}_C}) \triangleq \sum_{m=1}^{C} \sum_{k \in \mathcal{C}_m} J_k(w_k) \tag{3.7}$$

where $w_k^\circ = w_{\mathcal{C}_m}^\circ$. We collect the gradient vectors of the risk functions across the network into the aggregate vector

$$\nabla J(w) \triangleq \mathrm{col}\{\nabla J_1(w_1), \ldots, \nabla J_N(w_N)\}. \tag{3.8}$$

These gradients will not be available in most cases since the distribution of the data is not known to enable evaluation of the expectation of the loss functions. In stochastic-gradient implementations, it is customary to replace the above aggregate vector by the following approximation where the true gradients of the risk functions are replaced by

$$\widehat{\nabla J}(w) \triangleq \mathrm{col}\{\widehat{\nabla J_1}(w_1), \ldots, \widehat{\nabla J_N}(w_N)\} \tag{3.9}$$

where each $\widehat{\nabla J}_k(w_k)$ is constructed from the gradient of the respective loss function

$$\widehat{\nabla J}_k(w_k) \;=\; \nabla Q_k(w_k; \boldsymbol{z}_k) \tag{3.10}$$

evaluated at the corresponding data point, $\boldsymbol{z}_k$.

3.2.4 Assumptions

In this section, we list the assumptions that are needed to drive the analysis. These assumptions are typical in the analysis of stochastic-gradient algorithms, and most of them are automatically satisfied by important cases of interest, such as when the risk functions are quadratic or logistic — see, e.g., [22, 67].

We thus assume that each individual cost function $J_k(w_k)$ is twice-differentiable and τ_k-strongly convex [22, 24], for some $\tau_k > 0$. We also require the gradient vector of $J_k(w_k)$ to be ζ_k-Lipschitz, i.e.,

$$\|\nabla J_k(w_{k_1}) - \nabla J_k(w_{k_2})\| \leq \zeta_k \|w_{k_1} - w_{k_2}\| \tag{3.11}$$

for any $w_{k_1}, w_{k_2} \in \mathbb{R}^M$. Thus, the Hessian matrix function $\nabla^2 J_k(w_k)$ is bounded by

$$\tau_k I_M \leq \nabla^2 J_k(w_k) \leq \; \zeta_k I_M \tag{3.12}$$

where $\tau_k \leq \zeta_k$. Each Hessian matrix function is also assumed to satisfy the Lipschitz condition:

$$\|\nabla^2 J_k(w_{k_1}) - \nabla^2 J_k(w_{k_2})\| \leq \kappa_k \|w_{k_1} - w_{k_2}\| \tag{3.13}$$

for some $\kappa_k \geq 0$ and any $w_{k_1}, w_{k_2} \in \mathbb{R}^M$. The network gradient noise is denoted by $\boldsymbol{s}_i(\boldsymbol{w}_{i-1})$ which is the random process defined by

$$\boldsymbol{s}_i(\boldsymbol{w}_{i-1}) \triangleq \mathrm{col}\{\boldsymbol{s}_{1,i}(\boldsymbol{w}_{1,i-1}), \ldots, \boldsymbol{s}_{N,i}(\boldsymbol{w}_{N,i-1})\} \tag{3.14}$$

where the gradient noise at agent k at time instant i is given by

$$\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1}) \triangleq \widehat{\nabla J}_k(\boldsymbol{w}_{k,i-1}) - \nabla J_k(\boldsymbol{w}_{k,i-1}). \tag{3.15}$$

Here we are denoting the iterates $\boldsymbol{w}$ in boldface notation to indicate that they will actually be stochastic variables due to the approximation of the true gradients.

We let $\{\mathbb{F}_{k,i}; i \geq 0\}$ denote the filtration that collects all information up to time instant i. We then denote the conditional covariance matrix of $\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1})$ by

$$R_{k,i}(\boldsymbol{w}_{k,i-1}) \triangleq \mathbb{E}\left[\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1})\boldsymbol{s}_{k,i}^{\intercal}(\boldsymbol{w}_{k,i-1}) \mid \mathbb{F}_{k,i-1}\right]. \tag{3.16}$$

It is assumed that the gradient noise process satisfies the following properties for any $\boldsymbol{w}_{k,i-1}$ in $\mathbb{F}_{k,i-1}$ [22]:

1. Martingale difference [22, 67]:

$$\mathbb{E}\left[\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1}) \mid \mathbb{F}_{k,i-1}\right] = 0 \tag{3.17}$$

2. Bounded fourth-order moment [22, 67]:

$$\mathbb{E}\left[\|\boldsymbol{s}_{k,i}(\boldsymbol{w}_{k,i-1})\|^4 \mid \mathbb{F}_{k,i-1}\right] \leq \beta_k^2 \|w_k^\circ - \boldsymbol{w}_{k,i-1}\|^4 + \rho_k^4 \tag{3.18}$$

for some $\beta_k^2, \rho_k^4 \geq 0$.

3. Lipschitz conditional covariance function [22, 67]:

$$\|R_{k,i}(w_k^\circ) - R_{k,i}(\boldsymbol{w}_{k,i-1})\| \leq \theta_k \|w_k^\circ - \boldsymbol{w}_{k,i-1}\|^{\eta_k} \tag{3.19}$$

for some $\theta_k \geq 0$ and $0 < \eta_k \leq 4$.

4. Convergent conditional covariance matrix [22, 67]:

$$R_k \triangleq \lim_{i \to \infty} R_{k,i}(w_k^\circ) > 0 \tag{3.20}$$

where R_k is symmetric and positive definite.

3.2.5 Data Model

We assume that each agent k runs an independent stochastic gradient-descent algorithm of the form:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \widehat{\nabla J}_k(\boldsymbol{\psi}_{k,i-1}) \tag{3.21}$$

where $\mu_k > 0$ is a small step-size parameter, and $\boldsymbol{\psi}_{k,i}$ denotes the intermediate estimate for w_k° at time instant i. Cooperation among agents will be limited to neighbors that belong to the same cluster. Therefore, following the update (3.21), ideally, agent k should only share data with agent ℓ if $w_k^\circ = w_\ell^\circ$. The agents do not know which agents in their neighborhood belong to the same cluster; this information is learned in real-time. Therefore, agent k will only share data with agent ℓ if it believes that $w_k^\circ = w_\ell^\circ$. Specifically, agent k will combine the estimates from its neighbors in a convex manner as follows:

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i)\boldsymbol{\psi}_{\ell,i} \tag{3.22}$$

where the non-negative combination coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ satisfy

$$\boldsymbol{a}_{kk}(i) > 0,\ \boldsymbol{a}_{\ell k}(i) = 0 \ \text{ for } \ell \notin \boldsymbol{\mathcal{N}}_{k,i}, \quad \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) = 1. \tag{3.23}$$

In the next section, we explain how the combination coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ are selected in order to perform the combined tasks of estimation and clustering.

3.3 Clustering Scheme

Let $\delta > 0$ denote the smallest distance among the cluster models, $\{w_{\mathcal{C}_m}^\circ\}$. For any distinct $a, b \in \{1, \ldots, C\}$, it then holds that

$$\|w_{\mathcal{C}_a}^\circ - w_{\mathcal{C}_b}^\circ\| \geq \delta. \tag{3.24}$$

We introduce an $N \times N$ trust matrix $\boldsymbol{F}_i$; each entry $\boldsymbol{f}_{\ell k}(i) \in [0, 1]$ of this matrix reflects the amount of trust that agent k has in neighbor $\ell \in \mathcal{N}_k^-$ belonging to its cluster. The entries $\{\boldsymbol{f}_{\ell k}(i)\}$ are constructed as follows. Agent k first computes the Boolean variable:

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2 \leq \alpha, \\ 0, & \text{otherwise} \end{cases} \tag{3.25}$$

where α is a threshold value. The trust level $\boldsymbol{f}_{\ell k}(i)$ is smoothed as follows:

$$\boldsymbol{f}_{\ell k}(i) = \nu \boldsymbol{f}_{\ell k}(i-1) + (1-\nu)\boldsymbol{b}_{\ell k}(i) \tag{3.26}$$

where the forgetting factor, $0 < \nu < 1$, determines the speed with which trust in neighbor ℓ accumulates over time. Once $\boldsymbol{f}_{\ell k}(i)$ exceeds some threshold γ, agent k declares that neighbor ℓ belongs to its cluster and sets the corresponding entry $\boldsymbol{e}_{\ell k}(i)$ in matrix $\boldsymbol{E}_i$ to the value one:

$$\boldsymbol{e}_{\ell k}(i) = \begin{cases} 1, & \text{if } \boldsymbol{f}_{\ell k}(i) \geq \gamma, \\ 0, & \text{otherwise} \end{cases} \tag{3.27}$$

where $0 < \gamma < 1$. For completeness, we set for any agent k, $\boldsymbol{b}_{kk}(i) = \boldsymbol{f}_{kk}(i) = \boldsymbol{e}_{kk}(i) = 1$. Observe that the computation of the binary variable $\boldsymbol{b}_{\ell k}(i)$ couples the variables $\boldsymbol{\psi}$ and $\boldsymbol{w}$. Therefore, by using smoothed values $\{\boldsymbol{f}_{\ell k}(i)\}$ for the trust variables, we are able to couple the clustering and inference procedures into a single iterative algorithm rather than run them separately. The smoothing reduces the influence of erroneous clustering decisions on the inference task. The following listing summarizes the proposed strategy.

Algorithm 1 (Distributed clustering scheme)

Initialize $\boldsymbol{F}_{-1} = \boldsymbol{B}_{-1} = \boldsymbol{E}_{-1} = I$ and $\boldsymbol{\psi}_{-1} = \boldsymbol{w}_{-1} = 0$.
for $i \geq 0$ **do**
  for $k = 1, \ldots, N$ **do**

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \widehat{\nabla J_k}(\boldsymbol{\psi}_{k,i-1}) \tag{3.28}$$

  for $\ell \in \mathcal{N}_k^-$ **do**

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2 \leq \alpha \\ 0, & \text{otherwise} \end{cases} \tag{3.29}$$

$$\boldsymbol{f}_{\ell k}(i) = \nu \boldsymbol{f}_{\ell k}(i-1) + (1-\nu)\, \boldsymbol{b}_{\ell k}(i) \tag{3.30}$$

update $\boldsymbol{e}_{\ell k}(i)$ according to (3.27)

  end for

select $\{\boldsymbol{a}_{\ell k}(i)\}$ according to (3.23) and set

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) \boldsymbol{\psi}_{\ell,i} \tag{3.31}$$

  end for
end for

3.4 Mean-Square-Error Analysis

We now examine the mean-square performance of the proposed scheme.

We collect the estimates from across the network into the block vectors:

$$\boldsymbol{\psi}_i \triangleq \text{col}\{\boldsymbol{\psi}_{1,i}, \boldsymbol{\psi}_{2,i}, \ldots, \boldsymbol{\psi}_{N,i}\}, \tag{3.32}$$

$$\boldsymbol{w}_i \triangleq \text{col}\{\boldsymbol{w}_{1,i}, \boldsymbol{w}_{2,i}, \ldots, \boldsymbol{w}_{N,i}\}, \tag{3.33}$$

and define the matrices

$$\boldsymbol{\mathcal{A}}_i \triangleq \boldsymbol{A}_i \otimes I_M, \quad \mathcal{M} \triangleq \text{diag}\{\mu_1, \mu_2, \ldots, \mu_N\} \otimes I_M, \tag{3.34}$$

where $\boldsymbol{A}_i = [\boldsymbol{a}_{\ell k}(i)]$. From (3.21) we find that the network vector $\boldsymbol{\psi}_i$ evolves over time according to

$$\boldsymbol{\psi}_i = \boldsymbol{\psi}_{i-1} - \mathcal{M}\nabla J(\boldsymbol{\psi}_{i-1}) - \mathcal{M}\boldsymbol{s}_i(\boldsymbol{\psi}_{i-1}) \tag{3.35}$$

where $\nabla J(\cdot)$ and $\boldsymbol{s}_i(\cdot)$ are defined in (3.8) and (3.14). Likewise, from (3.22) we find that

$$\boldsymbol{w}_i = \boldsymbol{\mathcal{A}}_i^{\mathsf{T}}\boldsymbol{\psi}_i. \tag{3.36}$$

To proceed, we introduce the error vectors

$$\widetilde{\boldsymbol{\psi}}_{k,i} \triangleq w_k^{\circ} - \boldsymbol{\psi}_{k,i}, \quad \widetilde{\boldsymbol{w}}_{k,i} \triangleq w_k^{\circ} - \boldsymbol{w}_{k,i}, \tag{3.37}$$

and collect them from across the network into

$$\widetilde{\boldsymbol{\psi}}_i \triangleq \mathrm{col}\{\widetilde{\boldsymbol{\psi}}_{1,i}, \widetilde{\boldsymbol{\psi}}_{2,i}, \ldots, \widetilde{\boldsymbol{\psi}}_{N,i}\}, \tag{3.38}$$

$$\widetilde{\boldsymbol{w}}_i \triangleq \mathrm{col}\{\widetilde{\boldsymbol{w}}_{1,i}, \widetilde{\boldsymbol{w}}_{2,i}, \ldots, \widetilde{\boldsymbol{w}}_{N,i}\}. \tag{3.39}$$

We further define the network mean-square deviation (MSD) before and after the fusion step at the time instant i by

$$\mathrm{MSD}_{\psi}(i) \triangleq \mathbb{E}\,\|\widetilde{\boldsymbol{\psi}}_i\|^2, \tag{3.40}$$

$$\mathrm{MSD}_{w}(i) \triangleq \mathbb{E}\,\|\widetilde{\boldsymbol{w}}_i\|^2, \tag{3.41}$$

respectively.

3.4.1 Error Dynamics

Appealing to the mean-value theorem [22, p. 327] we can write

$$\nabla J(\boldsymbol{\psi}_{i-1}) = -\boldsymbol{\mathcal{H}}_{i-1}\widetilde{\boldsymbol{\psi}}_{i-1} \tag{3.42}$$

where

$$\boldsymbol{\mathcal{H}}_{i-1} \triangleq \mathrm{diag}\{\boldsymbol{H}_{k,i-1}\}_{k=1}^{N} \tag{3.43}$$

and each matrix $\boldsymbol{H}_{k,i-1}$ is given by

$$\boldsymbol{H}_{k,i-1} \triangleq \int_0^1 \nabla^2 J_k(w_k^{\circ} - t\widetilde{\boldsymbol{\psi}}_{k,i-1})dt. \tag{3.44}$$

Substituting (3.42) into (3.35) yields

$$\boldsymbol{\psi}_i = \boldsymbol{\psi}_{i-1} + \mathcal{M}\boldsymbol{\mathcal{H}}_{i-1}\widetilde{\boldsymbol{\psi}}_{i-1} - \mathcal{M}\boldsymbol{s}_i(\boldsymbol{\psi}_{i-1}). \tag{3.45}$$

By subtracting w° defined in (3.2) from both sides, we get

$$\widetilde{\boldsymbol{\psi}}_i = (I_{NM} - \mathcal{M}\boldsymbol{\mathcal{H}}_{i-1})\widetilde{\boldsymbol{\psi}}_{i-1} + \mathcal{M}\boldsymbol{s}_i(\boldsymbol{\psi}_{i-1}) \tag{3.46}$$

which means that the error recursion for each individual agent k is given by

$$\widetilde{\boldsymbol{\psi}}_{k,i} = (I_M - \mu_k \boldsymbol{H}_{k,i-1})\widetilde{\boldsymbol{\psi}}_{k,i-1} + \mu_k \boldsymbol{s}_{k,i}(\boldsymbol{\psi}_{k,i-1}). \tag{3.47}$$

It is argued in [22, p. 347] that for step-sizes μ_k satisfying

$$0 < \mu_k < \frac{2\tau_k}{\zeta_k^2 + \beta_k^2} \tag{3.48}$$

the mean-square-error quantity $\mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_{k,i}\|^2$ converges exponentially according to the recursion:

$$\mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_{k,i}\|^2 \leq \xi_k\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_{k,i-1}\|^2 + \mu_k^2 \rho_k^2 \tag{3.49}$$

where $0 \leq \xi_k < 1$ and is given by

$$\xi_k = 1 - 2\mu_k \tau_k + \mu_k^2(\zeta_k^2 + \beta_k^2). \tag{3.50}$$

It is further shown in [22, pp. 352, 378] that for small step-sizes satisfying (3.48), the error recursion (3.46) has bounded first, second, and fourth-order moments in the following sense:

$$\limsup_{i\to\infty}\ \|\mathbb{E}\ \widetilde{\boldsymbol{\psi}}_i\| = \mathcal{O}(\mu_{\max}) \tag{3.51}$$

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_i\|^2 = \mathcal{O}(\mu_{\max}) \tag{3.52}$$

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_i\|^4 = \mathcal{O}(\mu_{\max}^2) \tag{3.53}$$

where $\mu_{\max}$ is the maximum step-size across all agents.

We further introduce the constant block diagonal matrix:

$$\mathcal{H} \triangleq \operatorname{diag}\{H_1, H_2, \ldots, H_N\}, \quad H_k \triangleq \nabla^2 J_k(w_k^\circ), \tag{3.54}$$

and replace (3.46) by the approximate recursion

$$\widetilde{\boldsymbol{\psi}}_i' = (I_{NM} - \mathcal{M}\mathcal{H})\widetilde{\boldsymbol{\psi}}_{i-1}' + \mathcal{M}\boldsymbol{s}_i(\boldsymbol{\psi}_{i-1}) \tag{3.55}$$

where the random matrix $\boldsymbol{\mathcal{H}}_{i-1}$ is replaced by $\mathcal{H}$. It was also shown in [22, pp. 382, 384] that, for sufficiently small step-sizes, the error iterates that are generated by this recursion satisfy:

$$\lim_{i\to\infty}\ \mathbb{E}\ \widetilde{\boldsymbol{\psi}}_i' = 0 \tag{3.56}$$

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_i'\|^2 = \mathcal{O}(\mu_{\max}) \tag{3.57}$$

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_i - \widetilde{\boldsymbol{\psi}}_i'\|^2 = \mathcal{O}(\mu_{\max}^2) \tag{3.58}$$

$$\limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_i'\|^2 = \limsup_{i\to\infty}\ \mathbb{E}\ \|\widetilde{\boldsymbol{\psi}}_i\|^2 + \mathcal{O}(\mu_{\max}^{3/2}). \tag{3.59}$$

These results imply that, for large enough i, the errors $\widetilde{\boldsymbol{\psi}}$ and $\widetilde{\boldsymbol{\psi}}'$ are close to each other in the mean-square-error sense.

3.4.2 One Useful Property

The above construction guarantees one useful property if the clustering process does not incur errors of type II, meaning that links that should be disconnected are indeed disconnected. This implies that $w_\ell^\circ = w_k^\circ$ whenever $\boldsymbol{a}_{\ell k}(i) > 0$. Using (3.23), it follows that

$$\sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) w_\ell^\circ = w_k^\circ \tag{3.60}$$

or, equivalently,

$$\boldsymbol{\mathcal{A}}_i^{\mathsf{T}} w^\circ = w^\circ. \tag{3.61}$$

Subtracting w° from both sides of (3.36) yields

$$w^\circ - \boldsymbol{w}_i = w^\circ - \boldsymbol{\mathcal{A}}_i^{\mathsf{T}} \boldsymbol{\psi}_i. \tag{3.62}$$

Using (3.61) we rewrite (3.62) as:

$$\widetilde{\boldsymbol{w}}_i = \boldsymbol{\mathcal{A}}_i^{\mathsf{T}} \widetilde{\boldsymbol{\psi}}_i. \tag{3.63}$$

Taking the block maximum norm [7, p. 435] of both sides and using the sub-multiplicative property of norms implies that

$$\|\widetilde{\boldsymbol{w}}_i\|_{b,\infty} \le \|\boldsymbol{\mathcal{A}}_i^{\mathsf{T}}\|_{b,\infty} \, \|\widetilde{\boldsymbol{\psi}}_i\|_{b,\infty} = \|\widetilde{\boldsymbol{\psi}}_i\|_{b,\infty} \tag{3.64}$$

since $\boldsymbol{A}_i$ is left-stochastic and, therefore, $\|\boldsymbol{\mathcal{A}}_i^{\mathsf{T}}\|_{b,\infty} = 1$. It follows that

$$\mathbb{E}\, \|\widetilde{\boldsymbol{w}}_i\|_{b,\infty} \le \mathbb{E}\, \|\widetilde{\boldsymbol{\psi}}_i\|_{b,\infty}. \tag{3.65}$$

Results (3.64) and (3.65) ensure that the size of the error in the w domain is bounded by the size of the error in the ψ domain if there are no errors of type II during clustering.

3.5 Performance Analysis

We are ready to examine the behavior of the probabilities of erroneous decisions of types I and II for each agent k, namely, the probabilities that a link between k and one of its neighbors will be either erroneously disconnected (when it should be connected) or erroneously connected (when it should be disconnected):

$$\text{Type-I: } w_\ell^\circ = w_k^\circ \text{ and } \boldsymbol{a}_{\ell k}(i) = 0, \tag{3.66}$$

$$\text{Type-II: } w_\ell^\circ \neq w_k^\circ \text{ and } \boldsymbol{a}_{\ell k}(i) \neq 0 \tag{3.67}$$

for any $\ell \in \mathcal{N}_k$. After long enough i, these probabilities are denoted respectively by:

$$P_{\rm I} = \Pr\left(\boldsymbol{f}_{\ell k}(i) < \gamma \mid w_\ell^\circ = w_k^\circ\right), \tag{3.68}$$

$$P_{\rm II} = \Pr\left(\boldsymbol{f}_{\ell k}(i) \geq \gamma \mid w_\ell^\circ \neq w_k^\circ\right). \tag{3.69}$$

Assessing the probabilities (3.68) and (3.69) is a challenging task and needs to be pursued under some simplifying conditions to facilitate the analysis. This is due to the stochastic nature of the clustering and learning processes, and due to the coupling among the agents. Our purpose is to provide insights into the performance of these processes after sufficient learning time has elapsed. The analysis that follows adjusts the approach of [62] to the current setting. Different from [62] where it is assumed that there are only two models and all agents were trying to converge to one of these two models, we now have a multitude of clusters and agents that are trying to converge to their own cluster model.

3.5.1 Smoothing Process

In order to determine bounds for $P_{\rm I}$ and $P_{\rm II}$ we study the probability distribution of the trust variable $\boldsymbol{f}_{\ell k}(i)$. We have from (3.26) that:

$$\boldsymbol{f}_{\ell k}(i) = \nu^{i+1}\boldsymbol{f}_{\ell k}(-1) + (1-\nu)\sum_{j=0}^{i}\nu^j \boldsymbol{b}_{\ell k}(i-j) \tag{3.70}$$

where $\boldsymbol{b}_{\ell k}(i)$ is modelled as a Bernoulli random variable with success probability p:

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } (1-p). \end{cases} \tag{3.71}$$

We already know from (3.49) that, after sufficient time, the iterates $\boldsymbol{\psi}_{k,i}$ converge to the true models w_k° in the mean-square-error sense. Hence, it is reasonable to assume that the value of p becomes largely time-invariant and corresponds to the probability of the event described by

$$\|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2 \leq \alpha, \text{ for large } i. \tag{3.72}$$

We denote the probabilities of true and false assignments by

$$P_d = \Pr\left(\boldsymbol{b}_{\ell k}(i) = 1 \mid w_\ell^\circ = w_k^\circ\right), \tag{3.73}$$

$$P_f = \Pr\left(\boldsymbol{b}_{\ell k}(i) = 1 \mid w_\ell^\circ \neq w_k^\circ\right). \tag{3.74}$$

These probabilities also satisfy

$$(1-P_d) = \Pr\left(\|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2 > \alpha \mid w_\ell^\circ = w_k^\circ\right), \tag{3.75}$$

$$P_f = \Pr\left(\|\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}\|^2 \leq \alpha \mid w_\ell^\circ \neq w_k^\circ\right). \tag{3.76}$$

After a sufficient number of iterations, the influence of the initial condition in (3.70) can be ignored and we can approximate $\boldsymbol{f}_{\ell k}(i)$ by the following random geometric series:

$$\boldsymbol{f}_{\ell k}(i) \approx (1-\nu)\sum_{j=0}^{i} \nu^j \boldsymbol{b}_{\ell k}(i-j). \tag{3.77}$$

As explained in [62], although it is generally not true, we can simplify the analysis by assuming that, for large enough i, the random variables $\{\boldsymbol{b}_{\ell k}(m)\}$ in (3.77) are independent and identically distributed. This assumption is motivated by the fact that the models observed by the different clusters are assumed to be sufficiently distinct from each other by virtue of (3.24).

Now, recall that Markov's inequality [74, p. 47] implies that for any non-negative random variable $\boldsymbol{x}$ and positive scalar u, it holds that:

$$\Pr\left(\boldsymbol{x} \geq u\right) = \Pr\left(\boldsymbol{x}^2 \geq u^2\right) \leq \frac{\mathbb{E}\,\boldsymbol{x}^2}{u^2}. \tag{3.78}$$

To apply (3.78) to the variable $\boldsymbol{f}_{\ell k}(i)$, we need to determine its second-order moment. For this purpose, we follow [62] and introduce the change of variable:

$$\boldsymbol{b}_{\ell k}^\circ(i-j) \triangleq \frac{\boldsymbol{b}_{\ell k}(i-j) - p}{\sqrt{p(1-p)}}. \tag{3.79}$$

It can be verified that the variables $\{\boldsymbol{b}_{\ell k}^\circ(m)\}$ are i.i.d. with zero mean and unit variance. As a result, we rewrite (3.77) for large i as:

$$\boldsymbol{f}_{\ell k}(i) \approx p + \sqrt{p(1-p)}\,\boldsymbol{f}_{\ell k}^\circ(i) \tag{3.80}$$

where

$$\boldsymbol{f}_{\ell k}^\circ(i) \triangleq (1-\nu)\sum_{j=0}^{i} \nu^j \boldsymbol{b}_{\ell k}^\circ(i-j) \tag{3.81}$$

has zero mean and its variance is given by

$$\begin{aligned}\mathrm{Var}\left[\boldsymbol{f}_{\ell k}^\circ(i)\right] &= \mathbb{E}\left[\boldsymbol{f}_{\ell k}^\circ(i)\right]^2 - \left[\mathbb{E}\,\boldsymbol{f}_{\ell k}^\circ(i)\right]^2 \\ &= \frac{1-\nu}{1+\nu}(1-\nu^{2(i+1)}) \approx \frac{1-\nu}{1+\nu}.\end{aligned} \tag{3.82}$$

Returning to (3.68) we now have, with p replaced by P_d:

$$\begin{aligned} P_{\rm I} &\approx \Pr\left(\boldsymbol{f}_{\ell k}(i) < \gamma \mid w_\ell^\circ = w_k^\circ\right) \\ &\leq \Pr\left(|\boldsymbol{f}_{\ell k}^\circ(i)| > \frac{P_d - \gamma}{\sqrt{P_d(1-P_d)}} \;\middle|\; w_\ell^\circ = w_k^\circ\right) \\ &\leq \frac{1-\nu}{1+\nu} \cdot \frac{P_d(1-P_d)}{(P_d-\gamma)^2} \end{aligned} \tag{3.83}$$

where we applied (3.78) and the fact that, for any two generic events B_1 and B_2, if B_1 implies B_2, then the probability of event B_1 is less than the probability of event B_2 [75]. Similarly, by replacing p by P_f, we obtain

$$P_{\rm II} \leq \frac{1-\nu}{1+\nu} \cdot \frac{P_f(1-P_f)}{(\gamma-P_f)^2}. \tag{3.84}$$

In expressions (3.83) and (3.84), it is assumed that the size of the threshold value γ used in (3.27) satisfies $\gamma < P_d$ and $\gamma > P_f$. Since we usually desire the probability of false alarm to be small and the probability of detection to be close to one, these conditions can be met by $\gamma \in (0,1)$. We show in the next section that this is indeed the case.

Results (3.83) and (3.84) provide bounds on the probabilities of errors I and II. We next establish that $P_d \to 1$ and $P_f \to 0$ to conclude that $P_{\rm I} \to 0$ and $P_{\rm II} \to 0$.

3.5.2 The Distribution of the Variables

After a sufficient number of iterations and for small enough step-sizes, it is known that each $\boldsymbol{\psi}_{\ell,i}$ exhibits a distribution that is nearly Gaussian [32, 38, 76–79]:

$$\boldsymbol{\psi}_{\ell,i} \sim \mathbb{N}\left(w_\ell^\circ, \mu_\ell \Gamma_\ell\right) \tag{3.85}$$

where the matrix Γ_ℓ is symmetric, positive semi-definite, and the solution to the following Lyapunov equation [76]:

$$H_\ell \Gamma_\ell + \Gamma_\ell H_\ell = R_\ell \tag{3.86}$$

where the Hessian matrix H_ℓ is defined by (3.54) and R_ℓ is the steady-state covariance matrix of the gradient noise at agent ℓ defined by (3.20). We next introduce the vector

$$\bar{\boldsymbol{w}}_{k,i}^\circ \triangleq \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) w_\ell^\circ. \tag{3.87}$$

which should be compared with expression (3.22). The vector (3.87) is the result of fusing the actual models using the same combination weights available at time instant

i. It follows that $\boldsymbol{w}_{k,i}$ exhibits a distribution that is nearly Gaussian since the iterates $\{\boldsymbol{\psi}_{\ell,i}\}$ can be assumed to be independent of each other due to the decoupled nature of their updates:

$$\boldsymbol{w}_{k,i} \sim \mathbb{N}\,(\bar{\boldsymbol{w}}_{k,i}^{\circ}, \boldsymbol{\Omega}_{k,i}) \tag{3.88}$$

where $\boldsymbol{\Omega}_{k,i}$ is symmetric, positive semi-definite, and given by

$$\boldsymbol{\Omega}_{k,i} \triangleq \sum_{\ell=1}^{N} \mu_\ell\, \boldsymbol{a}_{\ell k}^2(i)\Gamma_\ell. \tag{3.89}$$

Let

$$\boldsymbol{g}_{\ell k,i} \triangleq \boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1} \tag{3.90}$$

and note that $\boldsymbol{g}_{\ell k,i}$ is again approximately Gaussian distributed with

$$\boldsymbol{g}_{\ell k,i} \sim \mathbb{N}\,(\bar{\boldsymbol{g}}_i, \boldsymbol{\Delta}_{\ell k,i}) \tag{3.91}$$

where

$$\bar{\boldsymbol{g}}_i \triangleq w_\ell^{\circ} - \bar{\boldsymbol{w}}_{k,i-1}^{\circ} \tag{3.92}$$

and $\boldsymbol{\Delta}_{\ell k,i}$ is symmetric, positive semi-definite, and bounded by (in view of Jensen's inequality [22, p. 769][2]):

$$\boldsymbol{\Delta}_{\ell k,i} \leq 2\,\big(\,\mu_\ell\Gamma_\ell + \boldsymbol{\Omega}_{k,i-1}\,\big). \tag{3.93}$$

From (3.89), (3.93) and for any ℓ and k it holds that:

$$\boldsymbol{\Delta}_{\ell k,i} = \mathcal{O}(\mu_{\max}). \tag{3.94}$$

3.5.3 The Statistics of $\|\boldsymbol{g}_{\ell k,i}\|^2$

We now examine the statistics of the main test variable for our algorithm from (3.25), namely, $\|\boldsymbol{g}_{\ell k,i}\|^2$. Let $\{\mathbb{A}_{k,i-1}; i > 0\}$ denote the filtration that collects all $\{\boldsymbol{a}_{\ell k}(i-1)\}$ information up to time instant $i-1$. Then, note that

$$\begin{aligned}\mathbb{E}\,\left[\|\boldsymbol{g}_{\ell k,i}\|^2 |\,\mathbb{A}_{k,i-1}\right] &= \mathbb{E}\,\left[\mathrm{Tr}\,\left(\boldsymbol{g}_{\ell k,i}\boldsymbol{g}_{\ell k,i}^{\mathsf{T}}\right)\big|\,\mathbb{A}_{k,i-1}\right] \\ &= \|\bar{\boldsymbol{g}}_i\|^2 + \mathrm{Tr}\,(\boldsymbol{\Delta}_{\ell k,i}).\end{aligned} \tag{3.95}$$

[2]Since $\mathbb{E}\,(\boldsymbol{a}+\boldsymbol{b})^2 \leq 2\mathbb{E}\,\boldsymbol{a}^2 + 2\mathbb{E}\,\boldsymbol{b}^2$.

Since $\boldsymbol{g}_{\ell k,i}$ is Gaussian, it holds that

$$\begin{aligned}
&\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}\|^4|\ \mathbb{A}_{k,i-1}\right] \\
&\quad=\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i+\bar{\boldsymbol{g}}_i\|^4|\ \mathbb{A}_{k,i-1}\right] \\
&\quad=\mathbb{E}\ \left[\left(\ \|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i\|^2+2(\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i)^{\mathsf{T}}\bar{\boldsymbol{g}}_i+\|\bar{\boldsymbol{g}}_i\|^2\right)^2\Big|\ \mathbb{A}_{k,i-1}\right] \\
&\quad=\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i\|^4|\ \mathbb{A}_{k,i-1}\right] \\
&\quad+2\mathbb{E}\left[\ \|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i\|^2\|\bar{\boldsymbol{g}}_i\|^2|\ \mathbb{A}_{k,i-1}\right]+\|\bar{\boldsymbol{g}}_i\|^4 \\
&\quad+4\bar{\boldsymbol{g}}_i^{\mathsf{T}}\mathbb{E}\left[\ (\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i)(\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i)^{\mathsf{T}}|\ \mathbb{A}_{k,i-1}\right]\bar{\boldsymbol{g}}_i \\
&\quad=\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i\|^4|\ \mathbb{A}_{k,i-1}\right]+2\ \mathrm{Tr}\ (\boldsymbol{\Delta}_{\ell k,i})\|\bar{\boldsymbol{g}}_i\|^2 \\
&\quad+\|\bar{\boldsymbol{g}}_i\|^4+4\|\bar{\boldsymbol{g}}_i\|^2_{\boldsymbol{\Delta}_{\ell k,i}}
\end{aligned}\tag{3.96}$$

where all odd order moments of $(\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i)$ are zero. Likewise,

$$\begin{aligned}
&\left(\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}\|^2|\ \mathbb{A}_{k,i-1}\right]\ \right)^2 \\
&\quad=\left(\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i+\bar{\boldsymbol{g}}_i\|^2|\ \mathbb{A}_{k,i-1}\right]\ \right)^2 \\
&\quad=\left(\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i\|^2|\ \mathbb{A}_{k,i-1}\right]+\|\bar{\boldsymbol{g}}_i\|^2\right)^2 \\
&\quad=[\ \mathrm{Tr}\ (\boldsymbol{\Delta}_{\ell k,i})\ ]^2+2\ \mathrm{Tr}\ (\boldsymbol{\Delta}_{\ell k,i})\|\bar{\boldsymbol{g}}_i\|^2 \\
&\quad+\|\bar{\boldsymbol{g}}_i\|^4.
\end{aligned}\tag{3.97}$$

According to Lemma A.2 of [5, p. 11], we have

$$\begin{aligned}
&\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}-\bar{\boldsymbol{g}}_i\|^4|\ \mathbb{A}_{k,i-1}\right] \\
&\quad=[\ \mathrm{Tr}\ (\boldsymbol{\Delta}_{\ell k,i})\ ]^2+2\ \mathrm{Tr}\ (\boldsymbol{\Delta}^2_{\ell k,i}).
\end{aligned}\tag{3.98}$$

Using (3.96) and (3.98), the variance of $\|\boldsymbol{g}_{\ell k,i}\|^2$ is given by

$$\begin{aligned}
&\mathrm{Var}\ \left[\|\boldsymbol{g}_{\ell k,i}\|^2|\ \mathbb{A}_{k,i-1}\right] \\
&\quad=4\|\bar{\boldsymbol{g}}_i\|^2_{\boldsymbol{\Delta}_{\ell k,i}}+2\ \mathrm{Tr}\ (\boldsymbol{\Delta}^2_{\ell k,i}).
\end{aligned}\tag{3.99}$$

Note from (3.95) that the mean of $\|\boldsymbol{g}_{\ell k,i}\|^2$ is dominated by $\|\bar{\boldsymbol{g}}_i\|^2$ for sufficiently small step-sizes. It follows from the Chebyshev's inequality [80, p. 455] that:

$$\begin{aligned}
&\Pr\left(\left|\|\boldsymbol{g}_{\ell k,i}\|^2-\mathbb{E}\ \left[\|\boldsymbol{g}_{\ell k,i}\|^2|\ \mathbb{A}_{k,i-1}\right]\ \right|\geq u\Big|\ \mathbb{A}_{k,i-1}\right) \\
&\quad\leq\frac{\mathrm{Var}\ \left[\|\boldsymbol{g}_{\ell k,i}\|^2|\ \mathbb{A}_{k,i-1}\right]}{u^2}=\mathcal{O}(\mu_{\max})
\end{aligned}\tag{3.100}$$

for any constant $u>0$, which implies that the variance of $\|\boldsymbol{g}_{\ell k,i}\|^2$ is in the order of $\mu_{\max}$. Therefore, when $w^\circ_\ell=w^\circ_k$ the probability mass of $\|\boldsymbol{g}_{\ell k,i}\|^2$ will concentrate around $\mathbb{E}(\|\boldsymbol{g}_{\ell k,i}\|^2)$, which is in the order of $\mathcal{O}(\mu_{\max})\approx 0$. On the other hand, when $w^\circ_\ell\neq w^\circ_k$, the probability mass of $\|\boldsymbol{g}_{\ell k,i}\|^2$ will concentrate around $\mathbb{E}(\|\boldsymbol{g}_{\ell k,i}\|^2)\approx\|\bar{\boldsymbol{g}}_i\|^2>0$. Obviously the threshold should be chosen as: $0<\alpha<\delta^2$, where δ is the clustering resolution.

3.5.4 Error Probabilities

It is seen from (3.75) and (3.76) that $1 - P_d$ corresponds to the right tail probability of $\|\boldsymbol{g}_{\ell k,i}\|^2$ for $w_\ell^\circ = w_k^\circ$, and P_f corresponds to the left tail probability of $\|\boldsymbol{g}_{\ell k,i}\|^2$ for $w_\ell^\circ \neq w_k^\circ$. To examine these probabilities, we follow arguments similar to [67] and apply them to the current context. We introduce the eigen-decomposition

$$\boldsymbol{\Delta}_{\ell k,i} = \boldsymbol{U}_i \boldsymbol{\Lambda}_i \boldsymbol{U}_i^{\mathsf{T}} \tag{3.101}$$

where $\boldsymbol{U}_i$ is orthonormal and $\boldsymbol{\Lambda}_i$ is diagonal and non-negative-definite. We further introduce the normalized variables:

$$\boldsymbol{x}_i \triangleq \boldsymbol{\Lambda}_i^{-1/2} \boldsymbol{U}_i^{\mathsf{T}} \boldsymbol{g}_{\ell k,i}, \tag{3.102}$$

$$\bar{\boldsymbol{x}}_i \triangleq \boldsymbol{\Lambda}_i^{-1/2} \boldsymbol{U}_i^{\mathsf{T}} \bar{\boldsymbol{g}}_i \tag{3.103}$$

and it follows from (3.91), (3.102), and (3.103) that

$$\boldsymbol{x}_i \sim \mathbb{N}\,(\bar{\boldsymbol{x}}_i, \mathbb{I}_M). \tag{3.104}$$

Note also from (3.102) that

$$\|\boldsymbol{g}_{\ell k,i}\|^2 = \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{\Lambda}_i \boldsymbol{x}_i = \sum_{h=1}^{M} \boldsymbol{\lambda}_{h,i} \boldsymbol{x}_{h,i}^2 \tag{3.105}$$

where $\boldsymbol{x}_{h,i}$ denotes the $h-$th element of $\boldsymbol{x}_i$ and $\boldsymbol{\lambda}_{h,i}$ denotes the $h-$th diagonal element of $\boldsymbol{\Lambda}_i$.

The probability $1 - P_d$:
It follows from the inequality

$$\|\boldsymbol{g}_{\ell k,i}\|^2 = \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{\Lambda}_i \boldsymbol{x}_i \leq \|\boldsymbol{\Delta}_{\ell k,i}\| \cdot \|\boldsymbol{x}_i\|^2, \tag{3.106}$$

that the following relation is satisfied

$$\{\|\boldsymbol{g}_{\ell k,i}\|^2 > \alpha\} \subseteq \{\|\boldsymbol{\Delta}_{\ell k,i}\| \cdot \|\boldsymbol{x}_i\|^2 > \alpha\}. \tag{3.107}$$

Defining

$$\boldsymbol{\alpha}_k(i) \triangleq \alpha / \|\boldsymbol{\Delta}_{\ell k,i}\|, \tag{3.108}$$

we can write using (3.107)

$$\Pr\left(\|\boldsymbol{g}_{\ell k,i}\|^2 > \alpha \mid w_\ell^\circ = w_k^\circ\right) \leq \Pr\left(\|\boldsymbol{x}_i\|^2 > \boldsymbol{\alpha}_k(i) \mid \bar{\boldsymbol{x}}_i = 0\right). \tag{3.109}$$

We know from (3.104) that

$$\|\boldsymbol{x}_i\|^2 \sim \mathcal{X}_M^2 \tag{3.110}$$

where $\mathcal{X}^2_M$ denotes the Chi-square distribution with M degrees of freedom and its mean value is M. According to the Chernoff bound for the central Chi-square distribution with M degrees of freedom[3] we have

$$\begin{aligned}
&\Pr\big(\ \|\boldsymbol{x}_i\|^2 > \boldsymbol{\alpha}_k(i)\ \big|\ \bar{\boldsymbol{x}}_i = 0\ \big) \\
&\quad = \Pr\ \left(\|\boldsymbol{x}_i\|^2 > \frac{M\cdot\boldsymbol{\alpha}_k(i)}{M}\ \middle|\ \bar{\boldsymbol{x}}_i = 0\right) \\
&\quad \le \exp\Big[-\frac{M}{2}\Big(\frac{\boldsymbol{\alpha}_k(i)}{M} - \log\Big(1+\frac{\boldsymbol{\alpha}_k(i)}{M}-1\Big)\Big)\Big] \\
&\quad = \Big(\frac{\boldsymbol{\alpha}_k(i)\cdot e}{M}\Big)^{M/2}\cdot\exp\Big[-\frac{\boldsymbol{\alpha}_k(i)}{2}\ \Big]
\end{aligned} \tag{3.111}$$

where e is Euler's number. For small enough step-sizes we conclude from (3.94), (3.108), and (3.111) that after a sufficient number of iterations, it holds that:

$$(1-P_d) \le \mathcal{O}(e^{-c_1/\mu_{\max}}) \tag{3.112}$$

for some constant $c_1 > 0$.

The probability P_f:
The approximate characteristic function of $\|\boldsymbol{g}_{\ell k,i}\|^2$ [67, Eq. (118)] for $w_\ell^\circ \neq w_k^\circ$ is given by

$$c_{\|\boldsymbol{g}_{\ell k,i}\|^2}(t) \approx e^{jt\|w_\ell^\circ - w_k^\circ\|^2 - 2t^2\|w_\ell^\circ - w_k^\circ\|^2_{\boldsymbol{\Lambda_i}}} \tag{3.113}$$

which implies that, for sufficiently small $\mu_{\max}$, we have

$$\|\boldsymbol{g}_{\ell k,i}\|^2 \sim \mathbb{N}\ (\|w_\ell^\circ - w_k^\circ\|^2, 4\|w_\ell^\circ - w_k^\circ\|^2_{\boldsymbol{\Lambda_i}}). \tag{3.114}$$

Therefore, from [67][4] we obtain that

$$\begin{aligned}
\Pr\ (\|\boldsymbol{g}_{\ell k,i}\|^2 < \alpha\ |w_\ell^\circ \neq w_k^\circ\ ) &\approx Q\left(\frac{\|w_\ell^\circ - w_k^\circ\|^2 - \alpha}{2\|w_\ell^\circ - w_k^\circ\|_{\boldsymbol{\Lambda_i}}}\right) \\
&\le \frac{1}{2}\exp\ \left[-\frac{(\|w_\ell^\circ - w_k^\circ\|^2 - \alpha)^2}{8\|w_\ell^\circ - w_k^\circ\|^2_{\boldsymbol{\Lambda_i}}}\right]
\end{aligned} \tag{3.115}$$

where the letter Q refers here to the traditional $Q-$function (the tail probability of the standard Gaussian distribution). For small enough step-sizes, after a sufficient number of iterations and from (3.94), (3.101), and (3.115), it holds that

$$P_f \le \mathcal{O}(e^{-c_2/\mu_{\max}}) \tag{3.116}$$

for some constant $c_2 > 0$. It is then seen that the probabilities $P_{\rm I}$ and $P_{\rm II}$ are expected to approach zero exponentially fast for vanishing step-sizes.

[3] Let $\boldsymbol{y} \sim \mathcal{X}^2_r$. According to the Chernoff bound for the central Chi-square distribution with r degrees of freedom, for any $\epsilon > 0$ it holds that [81, p. 2501]: $\Pr\ (\boldsymbol{y} > r(1+\epsilon)) \le \exp\ [-\frac{r}{2}(\epsilon - \log(1+\epsilon))]$.

[4] Let $\boldsymbol{y} \sim \mathbb{N}\ (0,1)$. According to the Chernoff bound for the Gaussian error function it holds that [82]: $Q(\boldsymbol{y}) \le \frac{1}{2}\exp\ [-\frac{\boldsymbol{y}^2}{2}]$.

3.6 Linking Application

3.6.1 Clustering With Linking Scheme

In this section we propose an additional mechanism to enhance the performance of each cluster by using the unused links to relay information. Figure 3.2 shows the linked topology that results for the same example shown earlier in Fig. 3.1(b). The figure shows that the links which are supposed to be unused for sharing data among neighbors belonging to different clusters, are now used to relay data among agents.

We assume in this section that the links among agents are *symmetric*, i.e., if $\ell \in \mathcal{N}_k \iff k \in \mathcal{N}_\ell$. Under normal operation, each agent k will be receiving and processing iterates only from those neighbors that it believes belong to the same cluster as k.

We modify this operation by allowing k to receive iterates from *all* of its neighbors. It will continue to use the iterates from neighbors in the same cluster to update its weight estimate $\boldsymbol{w}_{k,i}$. The iterates that arrive from neighbors that may belong to other clusters are not used during this fusion process. Instead, they will be relayed forward by agent k as follows. For each of its neighbors $\ell \in \mathcal{N}_k$ agent k will send $\boldsymbol{\psi}_{k,i}$ and another vector $\boldsymbol{\phi}_{k\ell,i}$. The vector $\boldsymbol{\phi}_{k\ell,i}$ is constructed as follows. Agent k chooses from among all the iterates it receives from its neighbors, that iterate that is closest to $\boldsymbol{\psi}_{\ell,i}$:

$$\boldsymbol{\phi}_{k\ell,i} = \underset{\substack{\{k,m\} \\ \forall m \in \mathcal{N}_k,\, m \notin \mathcal{N}_\ell}}{\arg\min} \|\boldsymbol{\psi}_{m,i} - \boldsymbol{\psi}_{\ell,i}\|^2. \tag{3.117}$$

Observe that the minimization is over k and all neighbors of k that are not neighbors of ℓ. This condition is important to avoid receiving the same information multiple times. Note that under this scheme, agent k will need to receive the iterates from all of its neighbors (those that it believes belong to its clusters and those that do not); it also needs to receive information about their neighborhoods, i.e., the $\mathcal{N}_\ell$ for each of its neighbors ℓ.

The following steps describe the clustering with linking algorithm. We collect all $\{\boldsymbol{\phi}_{\ell k,i}\}$ into a matrix $\boldsymbol{\Phi}_i$. By setting $\gamma = 0.5$ in Eq. (3.27) the operation of setting each entry $\boldsymbol{e}_{\ell k}(i)$ becomes rounding to the nearest integer and is denoted by $\lfloor \cdot \rceil$.

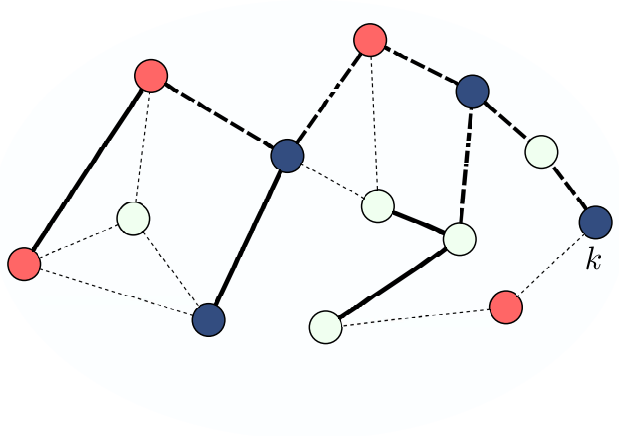

Figure 3.2. Clustered and linked topology that will result for the network shown in Fig. 3.1. The bold dashed lines depict the links used for relaying data among agents.

Algorithm 2 (Clustering with linking scheme)

Initialize $\boldsymbol{F}_{-1} = \boldsymbol{B}_{-1} = \boldsymbol{E} = I$, $\boldsymbol{\Phi}_{-1} = 0$, and $\boldsymbol{\psi}_{-1} = \boldsymbol{w}_{-1} = 0$.

for $i \geq 0$ **do**

for $k = 1, \ldots, N$ **do**

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \widehat{\nabla J_k}(\boldsymbol{\psi}_{k,i-1}) \tag{3.118}$$

for $\ell \in \mathcal{N}_k^-$ **do**

send $\boldsymbol{\psi}_{k,i}$ and $\boldsymbol{\phi}_{k\ell,i-1}$

receive $\boldsymbol{\psi}_{\ell,i}$ and $\boldsymbol{\phi}_{\ell k,i-1}$

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{\phi}_{\ell k,i-1} - \boldsymbol{w}_{k,i-1}\|^2 \leq \alpha \\ 0, & \text{otherwise} \end{cases} \tag{3.119}$$

$$\boldsymbol{f}_{\ell k}(i) = \nu \boldsymbol{f}_{\ell k}(i-1) + (1-\nu)\boldsymbol{b}_{\ell k}(i) \tag{3.120}$$

$$\boldsymbol{e}_{\ell k}(i) = \lfloor \boldsymbol{f}_{\ell k}(i) \rceil \tag{3.121}$$

end for

select $\{\boldsymbol{a}_{\ell k}(i)\}$ according to (3.23) and set

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i)\boldsymbol{\phi}_{\ell k,i-1} \tag{3.122}$$

update $\{\boldsymbol{\phi}_{k\ell,i}\}$ according to (3.117)

end for

end for

Agent k does not pick $\boldsymbol{\phi}_{\ell k,i-1}$ by itself, agent ℓ will choose it for k instead. Agent ℓ

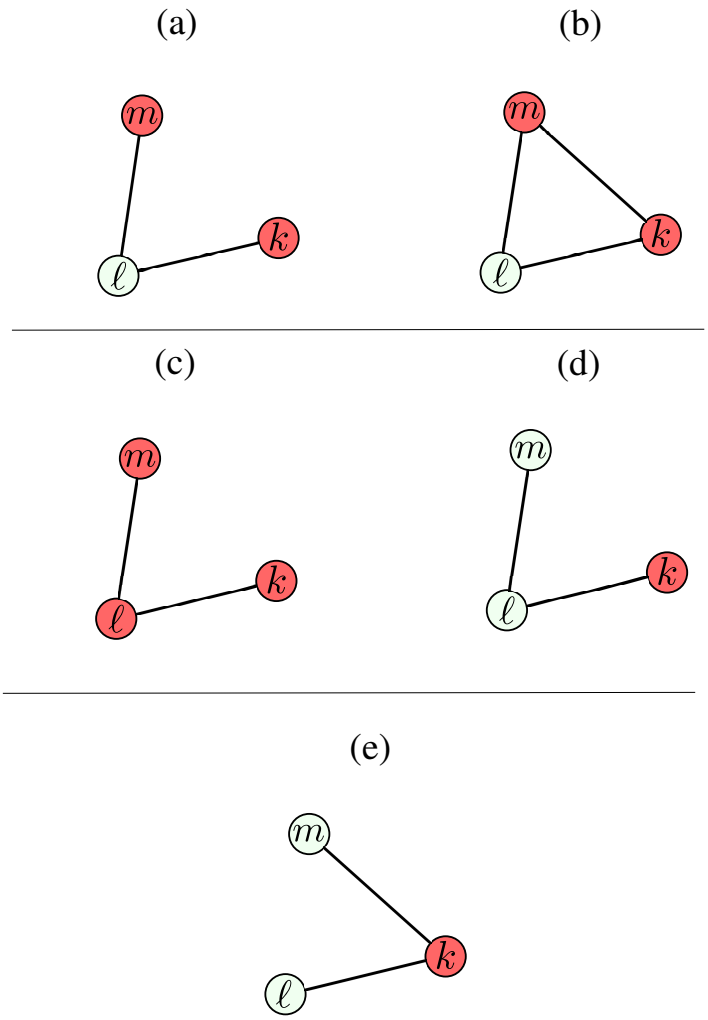

Figure 3.3. Examples of the linking situations. Solid links among agents depict the topology links, i.e., matrix Y.

needs $\boldsymbol{\psi}_{k,i}$ (to perform (3.117)) and needs the indices of k neighbors $\mathcal{N}_k$. $\mathcal{N}_k$ is needed to avoid receiving the same information multiple times. Thus, we assume the *symmetric* links among agents.

Note that in the linking scheme, agents need only one data exchange step during each iteration. That is the reason of using the time instant $(i-1)$ for $\boldsymbol{\phi}_{\ell k,i-1}$ through the algorithm scheme.

Examples:
Figure 3.3 shows some different examples to explain how the algorithm is working. After enough iterations (we drop the time index of the intermediate estimates and combination weights for simplicity), the situation of the links from agent ℓ's perspective and according to agent k converges as following:

(a) $\boldsymbol{a}_{\ell k} > 0$, because ℓ will pick $\boldsymbol{\psi}_m$ to send it as $\boldsymbol{\phi}_{\ell k}$. With time the trust between ℓ and k will be built (3.119) because $w_m^\circ = w_k^\circ$. That means, agent k will accept having data through agent ℓ.

(b) $\boldsymbol{a}_{\ell k} = 0$, where ℓ will *not* pick $\boldsymbol{\psi}_m$ to send it as $\boldsymbol{\phi}_{\ell k}$ because $m \in \mathcal{N}_k$. That means $\boldsymbol{\phi}_{\ell k} = \boldsymbol{\psi}_\ell$ (there are no other neighbors to pick their ψ's). This $\boldsymbol{\phi}_{\ell k}$ will always fail during the test (3.119) and will not build any trust between ℓ and k. That means agent k will not accept having any data through agent ℓ.

(c) $\boldsymbol{a}_{\ell k} > 0$, where ℓ picks either $\boldsymbol{\psi}_m$ or $\boldsymbol{\psi}_\ell$ to send it as $\boldsymbol{\phi}_{\ell k}$ because $w_m^\circ = w_k^\circ = w_\ell^\circ$.

(d) $\boldsymbol{a}_{\ell k} = 0$, because ℓ will pick either $\boldsymbol{\psi}_m$ or $\boldsymbol{\psi}_\ell$ to send it as $\boldsymbol{\phi}_{\ell k}$. This $\boldsymbol{\phi}_{\ell k}$ will always fail in the test (3.119).

(e) From agent k's perspective and according to agent ℓ: $\boldsymbol{a}_{k\ell} > 0$, because k will pick $\boldsymbol{\psi}_m$ to send it as $\boldsymbol{\phi}_{k\ell}$.
From agent k's perspective and according to agent m: $\boldsymbol{a}_{km} > 0$, because k will pick $\boldsymbol{\psi}_\ell$ to send it as $\boldsymbol{\phi}_{km}$.

3.6.2 Computational Complexity

From the steps of the algorithm, the computational complexity of the clustering scheme of agent k at each time instant is $\mathcal{O}(n_k)$. Likewise, the computational complexity of the clustering with linking scheme for agent k at each time instant is also $\mathcal{O}(n_k)$. Regarding the transmission complexity, the clustering with linking scheme requires double transmission lines. This is because each agent k sends two vectors $\boldsymbol{\psi}_{k,i}$ and $\boldsymbol{\phi}_{k\ell,i}$ to each neighbor ℓ instead of sending only one vector $\boldsymbol{\psi}_{k,i}$ in the clustering scheme case.

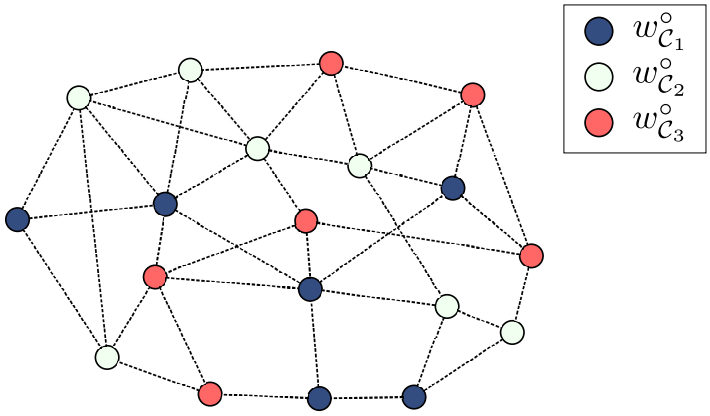

Figure 3.4. Network topology with connected clusters.

3.7 Master Election Application

From the fact that the total number of the models and their indices are not available by the agents, it is useful to have a distributed labeling system. The agents over the network do not know the indices of their observed models. In this section we propose a labeling system over multi-task networks by electing a master agent for each cluster. We assume that all agents belonging to the same cluster are connected. The master agent provides its index to label its cluster. Hence, we ensure that each cluster has a unique index.

Figure 3.4 shows the network structure where agents with the same color observe the same model. The unknown models are denoted by $\{w^{\circ}_{\mathcal{C}_1}, \ldots, w^{\circ}_{\mathcal{C}_C}\}$, each of size $M \times 1$.

During the clustering process, each cluster elects one master agent. The election criteria is that the agent with the smallest index in each cluster will be selected as the master agent of its cluster. Then, each master agent picks its index as a label of its observed model. Figure 3.5 shows the clustered topology and the master election result.

Each agent k denotes the index of the master agent of its cluster at time instant i by $\boldsymbol{n}_{\mathrm{m},k}(i)$. The initial value of $\boldsymbol{n}_{\mathrm{m},k}(i)$ at time instant $i = -1$ is given by

$$\boldsymbol{n}_{\mathrm{m},k}(-1) = k. \tag{3.123}$$

Then, agent k searches for the smallest index of the master agents that they are elected by $\boldsymbol{\mathcal{N}}_{k,i}$. If agent k finds a smaller index than its current master index, it will change its master index to this smaller one. Moreover, agent k denotes the index of the (source)

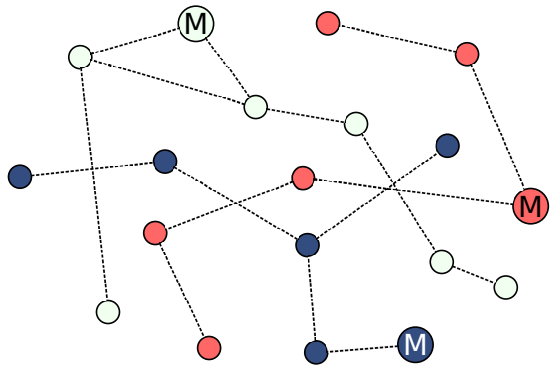

Figure 3.5. Clustered topology and the master election result.

agent that has the information about the smallest master index by $\boldsymbol{n}_{\mathrm{s},k}(i)$. Similarly, the initial value of $\boldsymbol{n}_{\mathrm{s},k}(i)$ at time instant $i = -1$ is given by

$$\boldsymbol{n}_{\mathrm{s},k}(-1) = k. \tag{3.124}$$

At each time instant i agent k verifies whether the source of its information still has the same master agent or if it is changed. In case it is changed, agent k will reset the value of the master index and the information source to the initial values again. Otherwise, if the source of the information still has the same master index as agent k, agent k searches again for the smallest index of the master agents that are elected by $\boldsymbol{\mathcal{N}}_{k,i}$. The election process occurs only within the clustered topology $\boldsymbol{\mathcal{N}}_{k,i}$.

Note that the algorithm tracks the drifters. If any failure occurs in the network structure, the agents will elect the new master agents in a proper and distributed way. Algorithm 3 summarizes the process of the master agent election.

3.8 Simulation Results

We consider a connected network with 50 randomly distributed agents. The agents observe data originating from three different models ($C = 3$). Each model $w^{\circ}_{\mathcal{C}_m} \in \mathbb{R}^{M\times 1}$ is generated as follows: $w^{\circ}_{\mathcal{C}_m} = [w_{r_1}, \ldots, w_{r_M}]^{\intercal}$, with entries $w_{r_c} \in [1, -1]$. In our example we set $M = 2$; larger values of M are generally easier for clustering and, therefore, we illustrate the operation of the algorithm for $M = 2$. The assignment of the agents to models is random. Agents having the same color belong to the same cluster. The maximum number of neighbors is $n_{\max} = 6$.

Every agent k has access to a scalar measurement $\boldsymbol{d}_k(i)$ and a $1 \times M$ regression vector $\boldsymbol{u}_{k,i}$. The measurements across the agents are assumed to be generated via the linear

Algorithm 3 (Master agent election scheme)

for $k = 1 : N$ **do**

$$\boldsymbol{p} = \boldsymbol{n}_{\mathrm{s},k}(i) \tag{3.125}$$

if $\boldsymbol{n}_{\mathrm{m},p}(i) = \boldsymbol{n}_{\mathrm{m},k}(i)$ **then**

for $\ell \in \boldsymbol{\mathcal{N}}_{k,i}$ **do**

if $\boldsymbol{n}_{\mathrm{m},k}(i) > \boldsymbol{n}_{\mathrm{m},\ell}(i)$ **then**

$$\boldsymbol{n}_{\mathrm{m},k}(i) = \boldsymbol{n}_{\mathrm{m},\ell}(i) \tag{3.126}$$

$$\boldsymbol{n}_{\mathrm{s},k}(i) = \ell \tag{3.127}$$

end if

end for

else

$$\boldsymbol{n}_{\mathrm{m},k}(i) = k \tag{3.128}$$

$$\boldsymbol{n}_{\mathrm{s},k}(i) = k \tag{3.129}$$

end if

end for

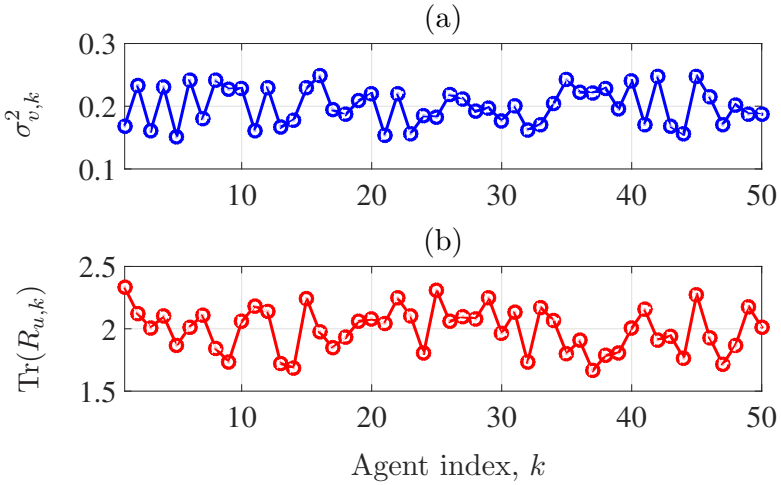

Figure 3.6. Statistical noise and signal profiles over the network.

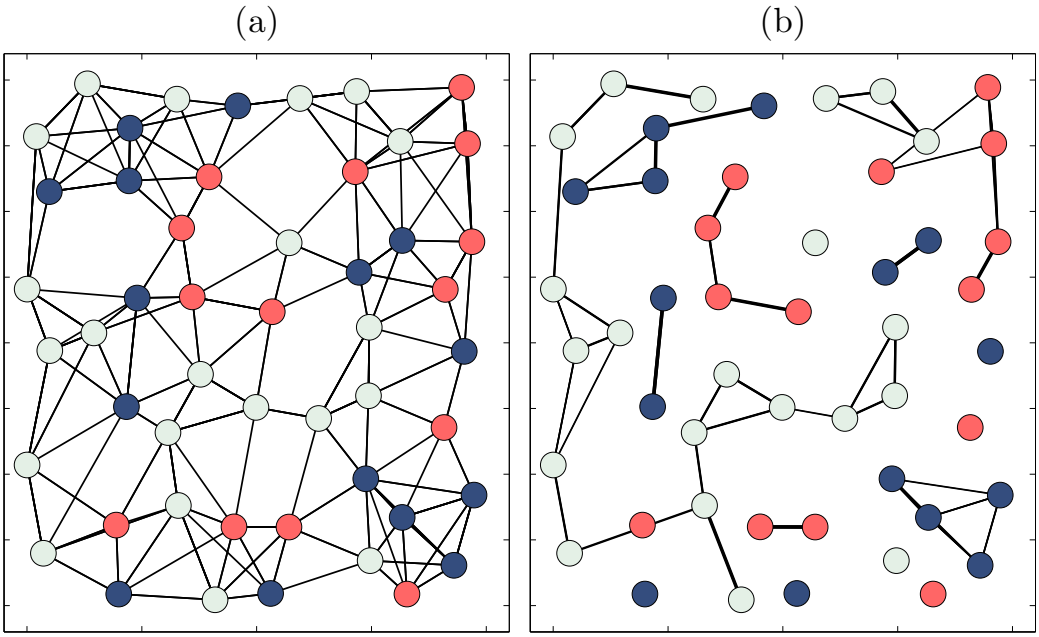

Figure 3.7. Network topology (a) and clustered topology at steady-state (b).

regression model

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w_k^{\circ} + \boldsymbol{v}_k(i) \tag{3.130}$$

where $\boldsymbol{v}_k(i)$ is measurement noise assumed to be a zero-mean white random process that is independent over space. It is also assumed that the regression data $\boldsymbol{u}_{k,i}$ is independent over space and independent of $\boldsymbol{v}_\ell(j)$ for all k, ℓ, i, j. All random processes are assumed to be stationary. The statistical profile of the noise across the agents for $k = 1, \ldots, N$ is shown in Fig. 3.6(a). The regressors are of size $M = 2$ and have diagonal covariance matrices $R_{u,k}$ as shown in Fig. 3.6(b). We set $\{\mu, \alpha, \nu, \delta, \gamma\} = \{0.05, 0.015, 0.98, 0.17, 0.5\}$. We use the uniform combination policy to generate the coefficients $\{a_{\ell k}(i)\}$.

3.8.1 Clustering Application

Figure 3.7(a) shows the topology of one of 100 Monte Carlo experiments. Figure 3.7(b) presents the final topology after applying the clustering technique. Figure 3.8(a) depicts the simulated transient mean-square deviation (MSD) of the network compared to other clustering methods. The model assignments change at time instant $i = 400$. The convergence of the curves shows the ability of the algorithm to track drifts in the models.

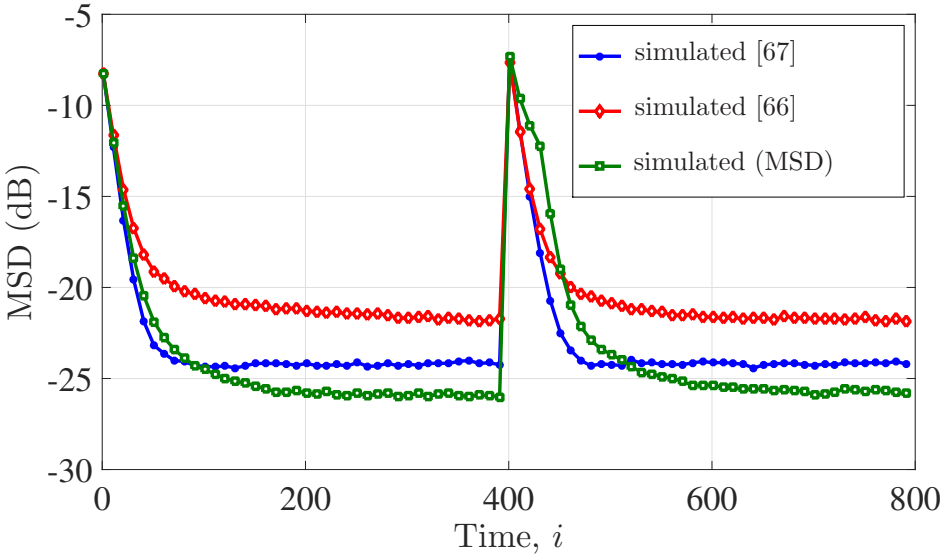

Figure 3.8. Transient mean-square deviation (MSD) using different approaches.

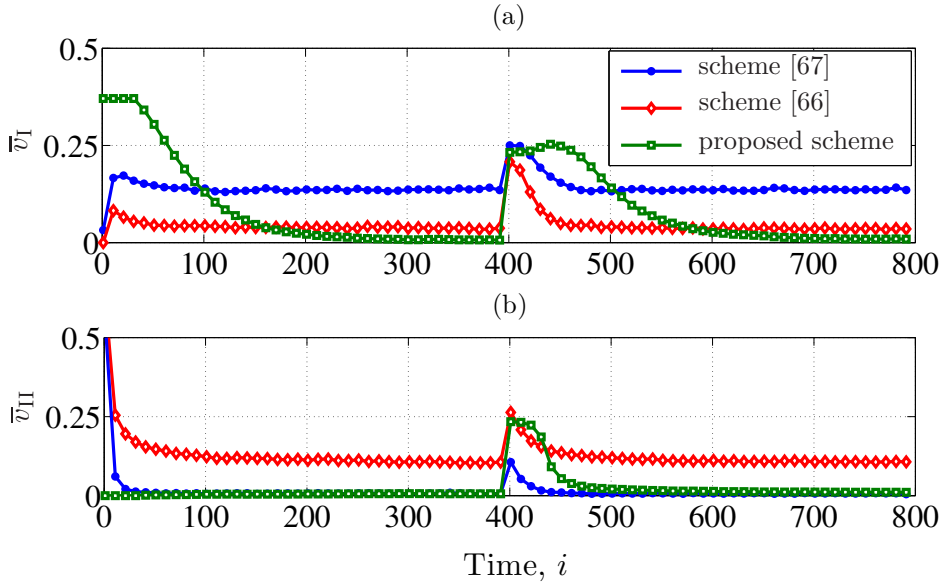

Figure 3.9. Normalized clustering errors of types I and II over the network.

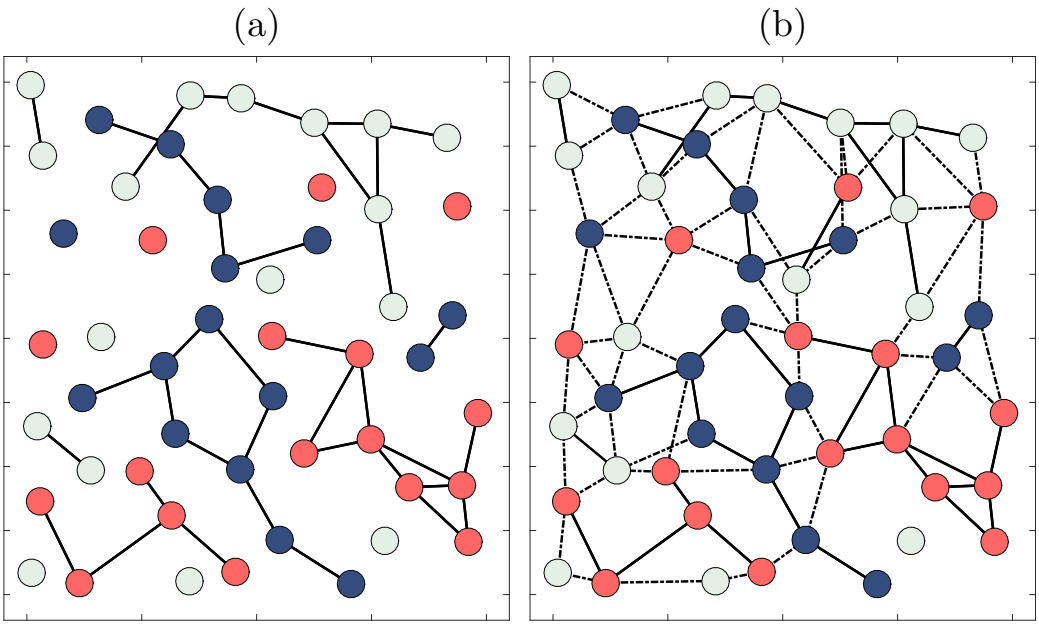

Figure 3.10. Clustered network topology at steady-state (a) and clustered and linked topology at steady-state (b).

The normalized clustering errors of types I and II by each agent k at time instant i are given, respectively, by

$$\boldsymbol{v}_{\mathrm{I},k}(i) \triangleq \frac{(\mathbb{1} - [\boldsymbol{E}_i]_{:,k})^{\intercal} \times ([E^{\circ}]_{:,k} - [\boldsymbol{E}_i]_{:,k})}{(n_k - 1)} \tag{3.131}$$

$$\boldsymbol{v}_{\mathrm{II},k}(i) \triangleq \frac{[\boldsymbol{E}_i]_{:,k}^{\intercal} \times ([\boldsymbol{E}_i]_{:,k} - [E^{\circ}]_{:,k})}{(n_k - 1)} \tag{3.132}$$

where E° is the true clustering matrix. Figures 3.9(a) and 3.9(b) depict the normalized clustering errors $\overline{v}_{\mathrm{I}}$ and $\overline{v}_{\mathrm{II}}$ over the network. The practical error probabilities go to zero at the steady-state.

3.8.2 Linking Application

Using the same setup of the previous example, Fig. 3.10(a) shows the topology of one experiment with the clustering technique only. Figure 3.10(b) presents the final topology when we apply the clustering with linking technique. Figure 3.11 indicates the simulated transient mean-square deviation (MSD) of the agents with and without the linking technique. The normalized clustering errors over the network are shown in Fig. 3.12.

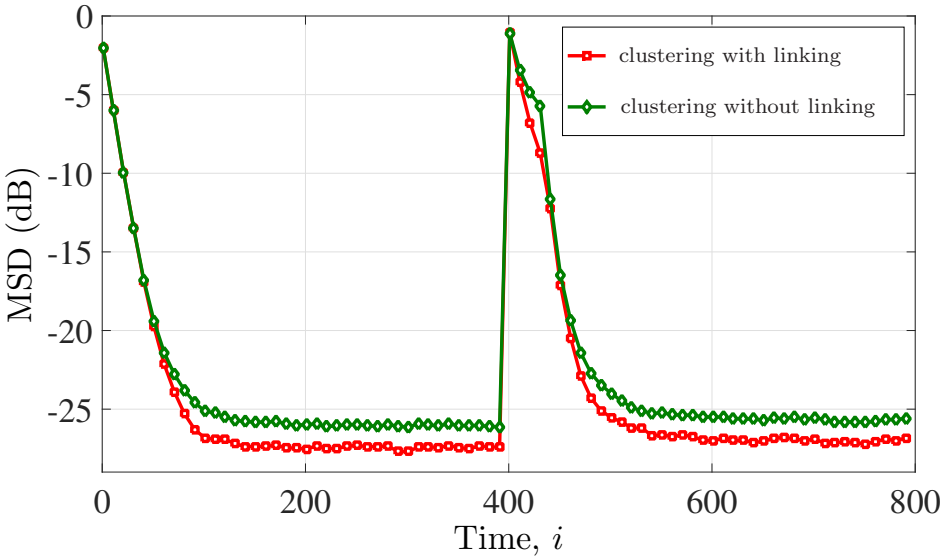

Figure 3.11. Transient mean-square deviation with and without applying the linking technique.

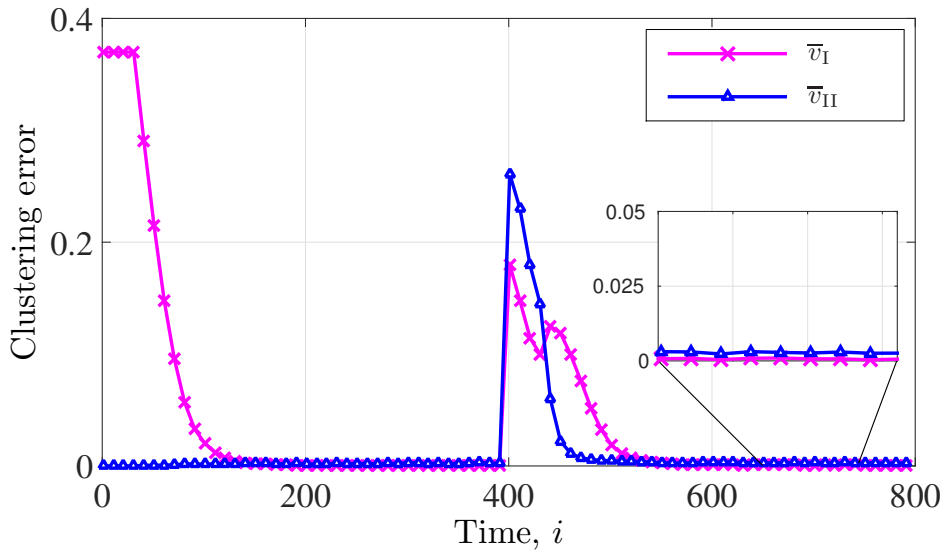

Figure 3.12. Normalized clustering errors of types I and II over the network.

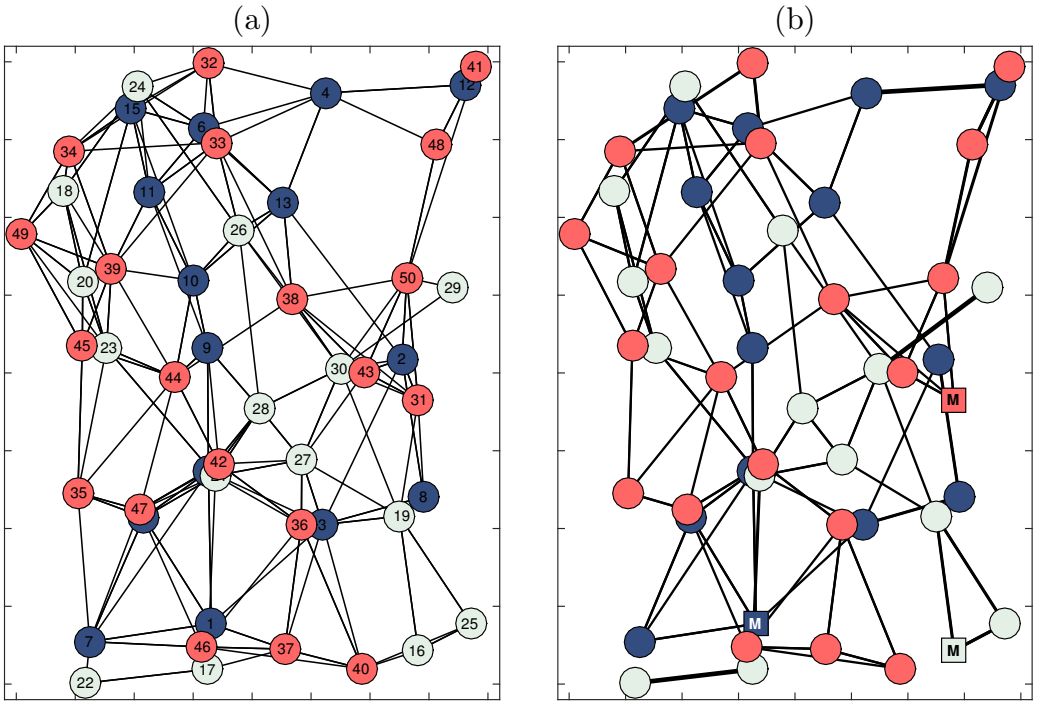

Figure 3.13. Network topology (a) and clustered topology with the master agents represented by squares (b).

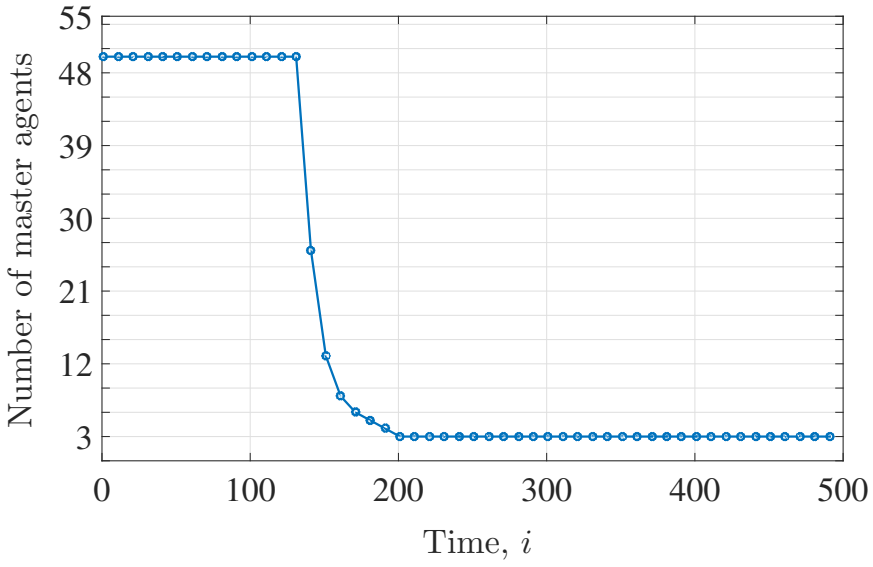

Figure 3.14. Number of the master agents in the network over time.

3.8.3 Master Election Application

We use the same setup of the previous example with the condition that the agents which belong to the same cluster are connected.

Figure 3.13(a) shows the topology of connected clusters. Figure 3.13(b) presents the final topology after applying the clustering technique and the master election process. Three master agents are selected and presented by squares. These three masters have the smallest indices among their cluster members.

Figure 3.14 depicts the number of the master agents in the network over time. This curve starts with N master agents because at time instant $i = -1$ all agents are master agents. The curve converges to the number of the observed model $C = 3$ at steady-state.

Chapter 4

Decentralized Partitioning Over Inhomogeneous Multi-Agent Networks

'Divide each difficulty into as many parts as is feasible and necessary to resolve it.'

René Descartes

Certain types of animal groups, such as bee swarms, consist of informed and uninformed agents where only the informed agents collect information about the environment. Moreover, in many situations agents can be subjected to data from different sources [30, 33, 42, 62, 63, 83].

In this chapter we consider a set of agents that are informed and observe different models. We propose a decentralized partitioning technique aimed at implementing a dynamic multi-task network using adaptation and learning in the presence of informed and uninformed agents. The algorithm ensures a fair partitioning process to distribute the uninformed agents among the groups that are interested in several objectives. Furthermore, the decentralized technique has a self-organizing feature that endows the network with a learning ability in stationary and non-stationary environments. We apply the proposed technique in both static and mobile networks and show that the size of the groups matches the centralized partitioning size well[1].

4.1 Introduction

Informed agents are defined as those agents that are capable of evaluating their gradient vector approximation continuously from streaming data and of performing the two tasks of adapting their iterates and consulting with their neighbors. Uninformed agents are incapable of performing adaptation but can still participate in the consultation process with their neighbors. The informed agents send information to the uninformed ones to

[1]This chapter is based on the conference paper: S. Khawatmi and A. M. Zoubir, "Decentralized partitioning over adaptive networks," *in Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, (Vietri sul Mare, Salerno, Italy), September 2016, pp. 1–6.

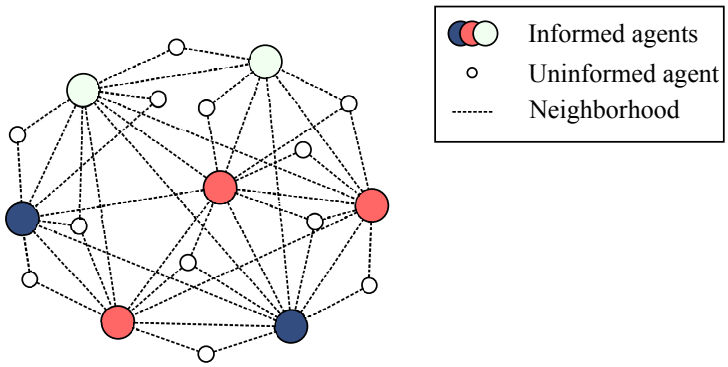

Figure 4.1. Network topology with three different clusters.

create a group supported by some uninformed agents. Each uninformed one responds to one informed agent and joins its group. In this case the first step is to cluster the informed agents. Then, groups are formed with respect to each model. To this end, we apply a clustering technique proposed in Sec. 3.3. The objective of the clustering step is to form groups of agents that observe the same model, share model information among one another, and ignore data from agents observing different models. After the informed agents cluster according to their respective models, each uninformed agent is assigned to one informed agent, joining its group and obtaining its model information.

In this chapter we formulate and solve the group allocation problem by designing the combination weights. The combination weights control the flow and exchange of data among agents to enable group formation. Our approach ultimately consolidates three processes: clustering, group formation, and estimation of the model parameters. We present some examples to illustrate our contributions. Furthermore, the algorithm is applied on mobile networks.

The chapter is organized as follows: we describe the network and data model in Sec. 4.2. Then, we illustrate in Sec. 4.3 the group forming technique. Finally, we present the simulation results in Sec. 4.4.

4.2 Network and Data Model

Consider a connected network consisting of N^{s} informed agents performing some tasks and sending information to the N^{r} uninformed agents ($N^{\mathrm{s}} \leq N^{\mathrm{r}}$). Each uninformed agent responds to one of the informed agents according to the information it receives.

The set of neighbors of agent k including k itself is denoted by $\mathcal{N}_k$. Two agents are considered neighbors if there is a direct connection between them. Figure 4.1 shows the network structure where agents with the same color observe the same model. We represent the network topology by means of the $N \times N$ adjacency matrix E, the entries $e_{\ell k}$ of which are defined as follows:

$$e_{\ell k} = \begin{cases} 1, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

Group $\mathcal{G}_{c,i}$ is the set of informed and uninformed agents that wish to estimate the same model parameters $w^{\circ}_{\mathcal{C}_c}$ at time instant i, where $c \in \{1, \ldots, C\}$ is the index of the observed model. It is assumed that $C \leq N^s$. In each group the agents perform the task of estimating a common observed model vector in real-time. The unknown models are each of size $M \times 1$ and given by

$$w^{\circ}_{\mathcal{C}} \triangleq \text{col}\{w^{\circ}_{\mathcal{C}_1}, w^{\circ}_{\mathcal{C}_2}, \ldots, w^{\circ}_{\mathcal{C}_C}\}. \tag{4.2}$$

The superscript $\circ$ is used to indicate true parameter values. At each time instant i, every informed agent k has access to a scalar measurement $\boldsymbol{d}_k(i)$ and a $1 \times M$ regression vector $\boldsymbol{u}_{k,i}$. The measurements across all agents are assumed to be generated via the linear regression model:

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w^{\circ}_{k,i} + \boldsymbol{v}_k(i). \tag{4.3}$$

All random processes are assumed to be stationary. Moreover, $\boldsymbol{v}_k(i)$ is zero-mean white measurement noise that is independent over space and has variance $\sigma^2_{v,k}$. It is assumed that the regression data $\boldsymbol{u}_{k,i}$ is a zero-mean Gaussian process, independent over time and space, and independent of $\boldsymbol{v}_\ell(j)$ for all k, ℓ, i, j. We denote the covariance matrix of $\boldsymbol{u}_{k,i}$ by $R_{u,k} \triangleq \mathbb{E}\boldsymbol{u}^{\mathsf{T}}_{k,i}\boldsymbol{u}_{k,i}$. The unknown models $\{w^{\circ}_{k,i}\}$ in Eq. (4.3) arise from the C models, i.e., $w^{\circ}_{k,i} = w^{\circ}_{\mathcal{C}_c}$ for some $c \in \{1, \ldots, C\}$ and $k = \{1, \ldots, N\}$. We stack the $w^{\circ}_{k,i}$ into a column vector:

$$w^{\circ}_i \triangleq \text{col}\{w^{\circ}_{1,i}, w^{\circ}_{2,i}, \ldots, w^{\circ}_{N,i}\}. \tag{4.4}$$

Note that the time index in $w^{\circ}_{k,i}$ is important for the uninformed agents because their desired models change over time according to the partitioning process. The general tasks of the whole system at each time instant are as follows: the informed agents observe different models and start to cluster themselves accordingly. They form groups and send information to pull the uninformed agents to their groups. The latter are each attracted to the informed agent exerting the strongest force.

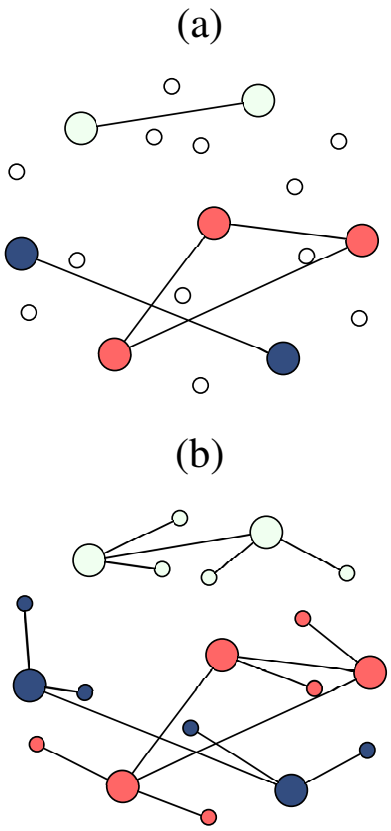

Figure 4.2. Clustered network (a) and group formation (b).

Figure 4.2 shows the clustered network and the groups in the network. Every informed agent is identified as being informed by all its neighbors by setting

$$n_{\text{in},k} = \begin{cases} 1, & k \in \mathcal{N}^{\text{s}}, \\ 0, & k \in \mathcal{N}^{\text{r}}. \end{cases} \tag{4.5}$$

where $n_{\text{in}} \triangleq \text{col}\{n_{\text{in},1}, n_{\text{in},2}, \ldots, n_{\text{in},N}\}$. From [18] each agent k runs the following LMS steps:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{\psi}_{k,i-1}) \tag{4.6}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) \boldsymbol{\psi}_{\ell,i} \tag{4.7}$$

where $\mu_k > 0$ is a small step-size and $\mu_k = 0$ when $k \in \mathcal{N}^{\text{r}}$. The current intermediate estimate of the parameter vector is denoted by $\boldsymbol{\psi}_{k,i}$ while $\boldsymbol{w}_{k,i}$ is the current estimate of the parameter vector. The non-negative combination coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ in Eq. (4.7) are seen to be time-dependent and should satisfy:

$$\boldsymbol{a}_{\ell k}(i) = 0 \quad \text{for } \ell \notin \boldsymbol{\mathcal{N}}_{k,i}, \ \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) = 1. \tag{4.8}$$

Observe that the combination coefficient $\boldsymbol{a}_{\ell k}(i)$ is non-zero only if agent k wishes to share data received from agent ℓ. The group information is retrieved from matrix $\boldsymbol{E}_i$ the entries of which are updated continuously according to

$$\boldsymbol{e}_{\ell k}(i) = \begin{cases} 1, & \ell \in \boldsymbol{\mathcal{N}}_{k,i}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

The coefficients $\{\boldsymbol{e}_{\ell k}(i)\}$ and $\{\boldsymbol{a}_{\ell k}(i)\}$ are selected as explained next.

4.3 Group Formation

Clustering is the process of computing the combination weights among the informed agents. Partitioning is the process of computing the combination weights among the informed and uninformed agents. The latter weights govern the data flow from the informed agents to the uninformed ones. Uninformed agents do not transmit data.

The uninformed agents adjust their combination weights in attempt to achieve the following goal: at steady-state all groups are required to have approximately the same size (fair partitioning), i.e., as

$$|\mathcal{G}_{1,i}| \approx |\mathcal{G}_{2,i}| \approx \ldots \approx |\mathcal{G}_{C,i}| \quad \text{as} \quad i \to \infty. \tag{4.10}$$

The size of each group is determined by the entries of the matrix $\boldsymbol{A}_i$.

4.3.1 Clustering Scheme

The informed agents try to collectively enhance the inference performance of the whole network in a distributed and cooperative manner. We apply the clustering technique proposed in Sec. 3.3 to create the estimated clustering matrix $\boldsymbol{F}_i$ of size $N \times N$ as follows. We initialize $\boldsymbol{\psi}_{k,-1} = 0$ and $\boldsymbol{B}_{-1} = \boldsymbol{F}_{-1} = \boldsymbol{E}_{-1} = I_N$. Each entry $\boldsymbol{a}_{\ell k}(i)$ is designed using the clustering algorithm proposed in Sec. 3.3, where $k, \ell \in \mathcal{N}^{\mathrm{s}}$:

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{if } ||\boldsymbol{\psi}_{\ell,i} - \boldsymbol{w}_{k,i-1}||^2 \leq \alpha, \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

$$\boldsymbol{f}_{\ell k}(i) = \nu \times \boldsymbol{f}_{\ell k}(i-1) + (1-\nu) \times \boldsymbol{b}_{\ell k}(i) \tag{4.12}$$

$$\boldsymbol{e}_{\ell k}(i) = \lfloor \boldsymbol{f}_{\ell k}(i) \rceil \tag{4.13}$$

where $\alpha > 0$, $0 \leq \nu \leq 1$, and the notation $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

In the following, we will introduce some quantities that have to be estimated in order to accomplish the partitioning process. For each agent $k \in \mathcal{N}^{\mathrm{s}}$ the estimated number of the informed agents within its group at time instant i is given by

$$\boldsymbol{g}_{\mathrm{in},k}(i) = [\boldsymbol{E}_i]_{:,k}^{\mathsf{T}} \mathbb{1}. \tag{4.14}$$

The estimated number of groups at time instant i by agent $k \in \mathcal{N}^{\mathrm{s}}$ is given by $\boldsymbol{g}_{\mathrm{gr},k}(i)$. To estimate the number of groups in the network, each agent k counts how many different estimation vectors $w_{\ell,i}$ its neighbors $\ell \in \mathcal{N}_k$ have. The estimated number of the desired uninformed agents by agent $k \in \mathcal{N}^{\mathrm{s}}$ is given by

$$\boldsymbol{n}_{\mathrm{z},k}(i) = \left\lfloor \frac{N - |\mathcal{N}^{\mathrm{s}} \cap \mathcal{N}_k|}{\boldsymbol{g}_{\mathrm{gr},k}(i) \times \boldsymbol{g}_{\mathrm{in},k}(i)} \right\rceil. \tag{4.15}$$

The actual number of uninformed agents that are following agent $k \in \mathcal{N}^{\mathrm{s}}$ is given by

$$\boldsymbol{n}_{\mathrm{f},k}(i) = [\boldsymbol{E}_i]_{k,:}(\mathbb{1} - n_{\mathrm{in},i}). \tag{4.16}$$

Therefore, the estimated number of needed uninformed agents by agent k at time instant i is given by

$$\boldsymbol{g}_{\mathrm{e},k}(i) = \boldsymbol{n}_{\mathrm{z},k}(i) - \boldsymbol{n}_{\mathrm{f},k}(i). \tag{4.17}$$

In case $\boldsymbol{g}_{\mathrm{e},k}(i) < 0$, agent k has too many uninformed followers.

4.3.2 Partitioning Scheme

Each uninformed agent decides which informed agent it will follow and receive information from. Matrix $\boldsymbol{A}_i$ determines the data flow among agents. Uninformed agents alter their group membership adaptively until convergence. Ultimately, each agent k will belong to one group. Consider an $N^{\mathrm{s}} \times N$ matrix $\boldsymbol{S}_i$ with non-negative real entries $\boldsymbol{s}_{\ell k}(i)$, each representing the force with which the informed agent ℓ pulls the uninformed agent k. Every entry $\boldsymbol{s}_{\ell k}(i)$ is given by

$$\boldsymbol{s}_{\ell k}(i) = \begin{cases} \frac{\boldsymbol{g}_{\mathrm{e},\ell}(i)}{\sum_{n \in \mathcal{N}_k \cap \mathcal{N}^{\mathrm{s}}} |\boldsymbol{g}_{\mathrm{e},n}(i)|}, & \ell \in \mathcal{N}^{\mathrm{s}} \cap \mathcal{N}_k, \\ 0, & \text{otherwise.} \end{cases} \tag{4.18}$$

Each entry $\boldsymbol{f}_{\ell k}(i)$ of the matrix $\boldsymbol{F}_i$, where $\ell \in \mathcal{N}^{\mathrm{s}}$ and $k \in \mathcal{N}^{\mathrm{r}}$, is adjusted using the following relation:

$$\boldsymbol{f}_{\ell k}(i) = (1 - \boldsymbol{b}_{\ell k}(i))\boldsymbol{e}_{\ell k}(i-1) + \boldsymbol{b}_{\ell k}(i)\boldsymbol{s}_{\ell k}(i). \tag{4.19}$$

The next step is to clip the negative values at zero:

$$\boldsymbol{f}_{\ell k}(i) = \begin{cases} \boldsymbol{f}_{\ell k}(i), & \text{if } \boldsymbol{f}_{\ell k}(i) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.20}$$

In Eq. (4.19) the factor $\boldsymbol{b}_{\ell k}(i)$ ensures fair partitioning and solves the oscillation problem. The latter occurs when there is an equilibrium state between two forces impacting the same agent. This problem hinders convergence. $\boldsymbol{b}_{\ell k}(i)$ is given as

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{if } \boldsymbol{g}_{\mathrm{e},\ell}(i) > 0 \text{ AND } \boldsymbol{e}_{\ell k}(i-1) = 0 \text{ AND (4.22)}, \\ \lambda, & \text{if } \boldsymbol{g}_{\mathrm{e},\ell}(i) < 0 \text{ AND } \boldsymbol{e}_{\ell k}(i-1) = 1 \text{ AND (4.23)}, \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

where $0 < \lambda < 1$. For $n \in \mathcal{N}^{\mathrm{s}} \cap \mathcal{N}_k$ conditions (4.22) and (4.23) are defined as follows:

$$\{\exists n : \boldsymbol{e}_{nk}(i-1) = 1 \text{ AND } \boldsymbol{g}_{\mathrm{e},n}(i) < 0\} \text{ OR } \{\boldsymbol{e}_{kk}(i-1) = 1\}. \tag{4.22}$$

The idea behind this condition is that agent ℓ may try to pull agent k in two cases: either if agent k is not following any agent yet, i.e., $\boldsymbol{e}_{kk}(i-1) = 1$, or when agent n which agent k follows has too many followers, i.e., $\boldsymbol{g}_{\mathrm{e},n}(i) < 0$. Condition (4.23) is given by

$$\{\exists n : \boldsymbol{e}_{nk}(i-1) = 0 \text{ AND } \boldsymbol{g}_{\mathrm{e},n}(i) > 0\}. \tag{4.23}$$

If $\boldsymbol{g}_{\mathrm{e},\ell}(i) < 0$ AND $\boldsymbol{e}_{\ell k}(i-1) = 1$, agent ℓ will not push agent k directly, but rather smooths the value with the previous $\boldsymbol{e}_{\ell k}(i-1)$. Each entry $\boldsymbol{e}_{\ell k}(i)$ of matrix $\boldsymbol{E}_i$ representing group membership is adjusted using the following relation:

$$\boldsymbol{e}_{\ell k}(i) = \begin{cases} 1, & \text{if } \boldsymbol{f}_{\ell k}(i) = \max([\boldsymbol{F}_i]_{:,k}) \text{ AND } \boldsymbol{g}_{\mathrm{e},\ell}(i) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.24}$$

In case two (or more) $\boldsymbol{f}_{\ell_1 k}(i) = \boldsymbol{f}_{\ell_2 k}(i) = \max([\boldsymbol{F}_i]_{:,k})$, the uninformed agent k randomly chooses only one of them to follow.

4.4 Simulation Results

4.4.1 Static Network

We consider a network with 40 randomly distributed agents. Figure 4.3 shows the statistical profile of the regressors and noise across the informed agents. The regressors are of size $M = 2$ as well as zero-mean Gaussian, independent in time and space, and have diagonal covariance matrices $R_{u,k}$. We chose $\{\alpha, \mu, \nu, \lambda\} = \{0.02, 0.05, 0.02, 0.6\}$, and $|\mathcal{N}_k| = 20$. We impose a priority for informed agents to be assigned as neighbors even if some other uninformed agents are closer, as long as they are within the radius $r_{\mathrm{nei}} = 2$. The first 10 agents are informed and denoted by big circles. Agents having

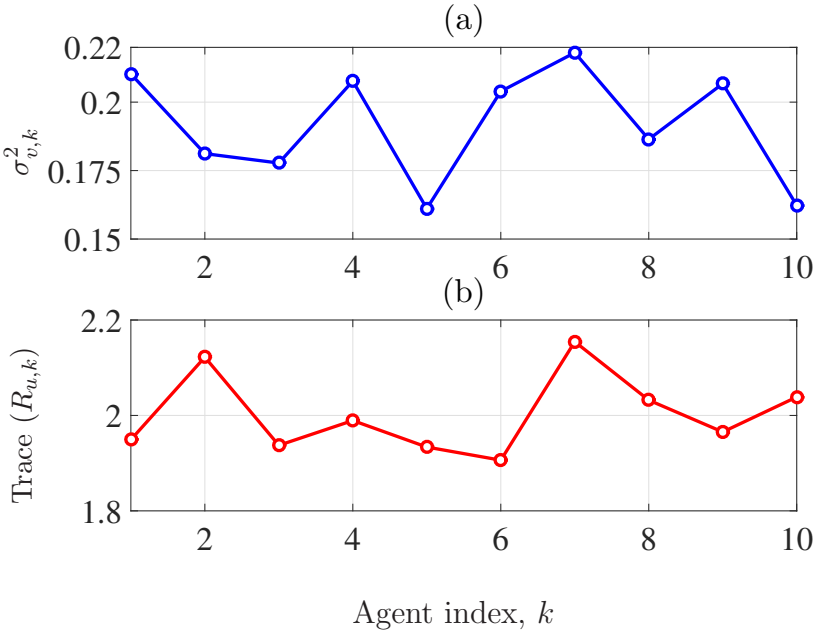

Figure 4.3. Statistical profiles of the informed agents.

the same color belong to the same group. The informed agents observe data originating from three different models $C = 3$. Each model $w^{\circ}_{\mathcal{C}_c} \in \mathbb{R}^{M\times 1}$ is generated as follows: $w^{\circ}_{\mathcal{C}_c} = [w_{r_1}, \ldots, w_{r_M}]^{\intercal}$ where $w_{r_m} \in [1, -1]$. The assignment of agents to models is random. We use a uniform combination policy to generate the coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$.

The simulation results are obtained by averaging over 1000 independent Monte Carlo runs. Figure 4.4 shows the topology of one of these experiments and the final structure after clustering and partitioning. Figure 4.5(a) depicts the transient mean-square deviation (MSD) of informed agents at each time instant i ginen by

$$\mathrm{MSD}(i) \triangleq \frac{1}{N^{\mathrm{s}}} \sum_{k\in\mathcal{N}^{\mathrm{s}}} \mathbb{E}\|w^{\circ}_k - \boldsymbol{w}_{k,i}\|^2. \tag{4.25}$$

Figure 4.5(b) depicts the estimated number of groups over time. The normalized clustering errors over the network are presented in Figure 4.5(c). The respective normalized clustering errors of each agent k at time instant i are given by

$$\boldsymbol{v}_{1,k}(i) \triangleq \frac{(\mathbb{1} - [\boldsymbol{E}_i]_{:,k})^{\intercal} \times ([E^{\circ}]_{:,k} - [\boldsymbol{E}_i]_{:,k})}{(n_k - 1)} \tag{4.26}$$

$$\boldsymbol{v}_{2,k}(i) \triangleq \frac{[\boldsymbol{E}_i]^{\intercal}_{:,k} \times ([\boldsymbol{E}_i]_{:,k} - [E^{\circ}]_{:,k})}{(n_k - 1)} \tag{4.27}$$

where n_k is the number of agent k's informed neighbors and E° is the true clustering matrix. Figure 4.5(d) shows the estimated group size where fair partitioning leads to 10 uninformed agents per group.

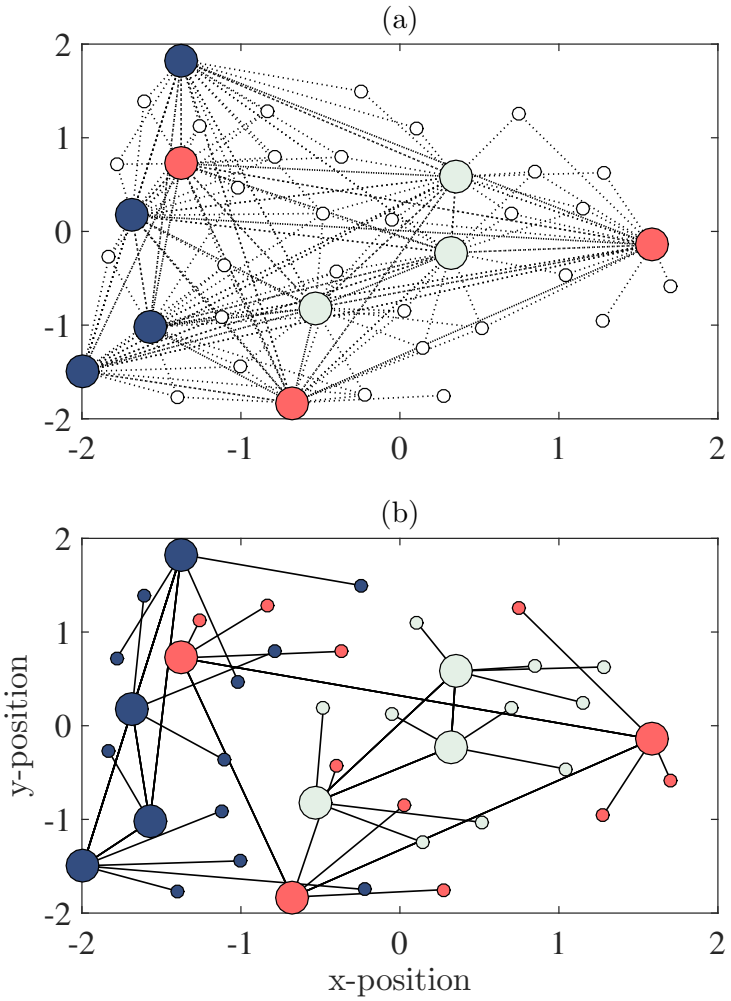

Figure 4.4. Network topology before (a) and after group formation (b), $N = 40$.

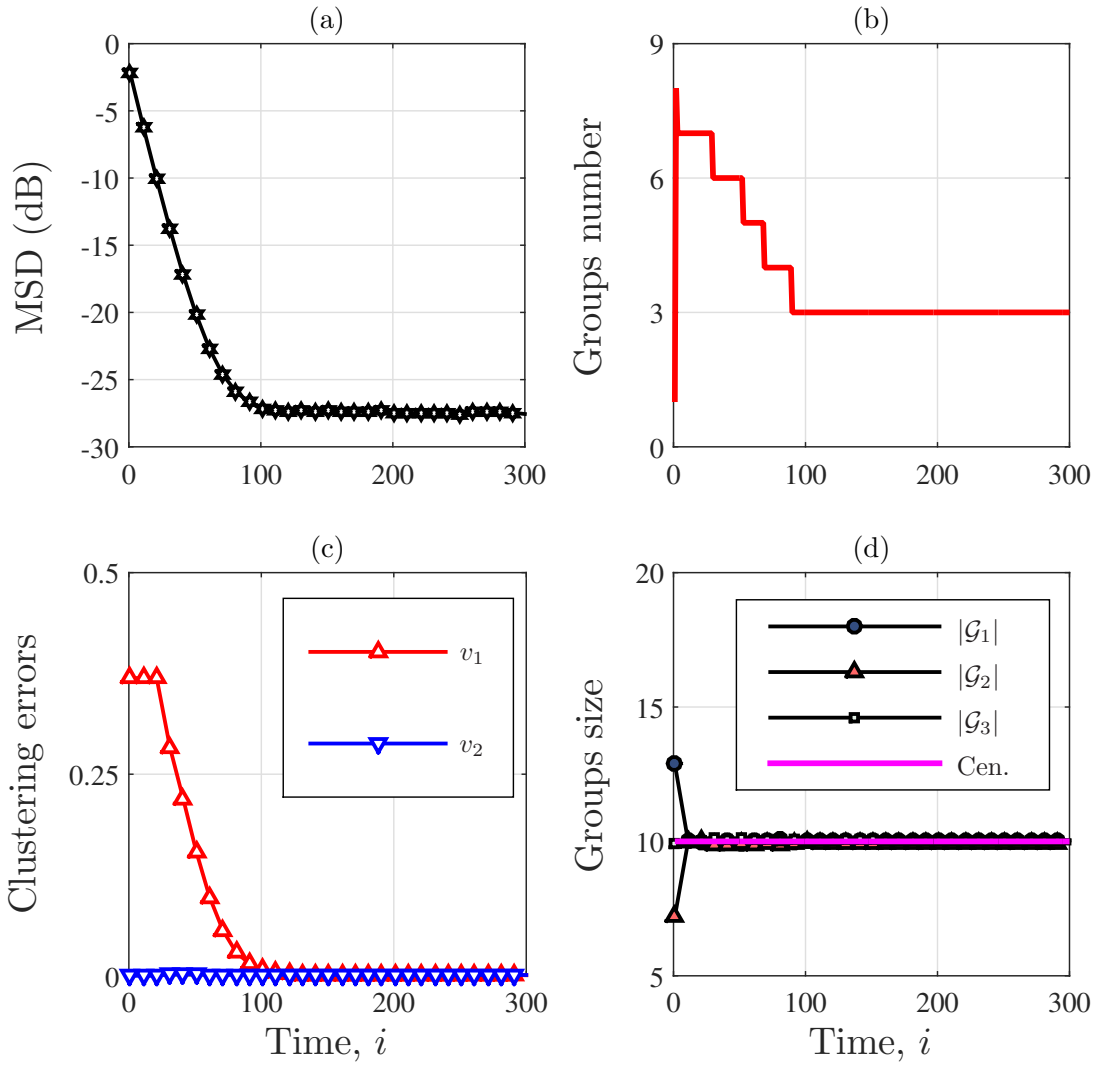

Figure 4.5. Network MSD (a), estimated number of groups (b), network clustering errors (c), and estimated group size (d).

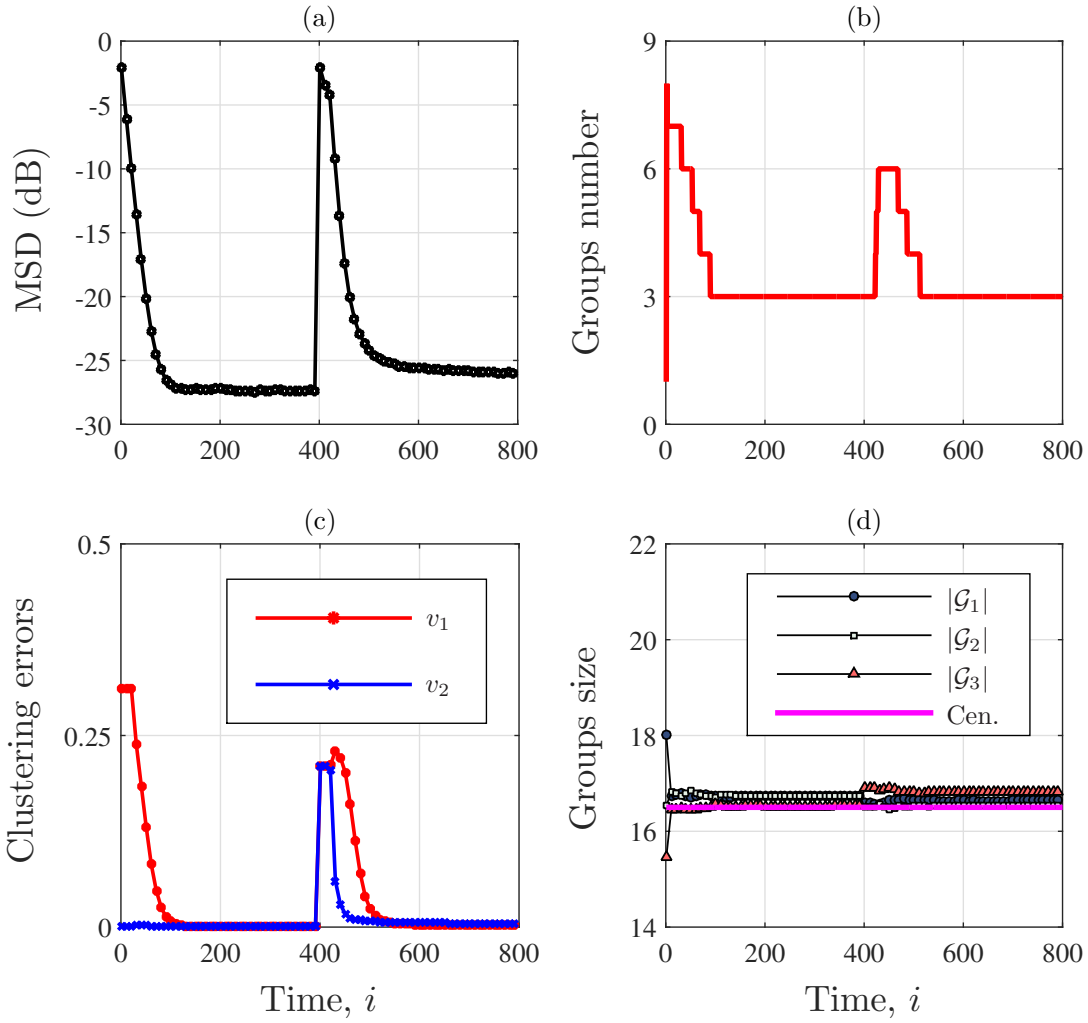

Figure 4.6. Network MSD (a), estimated number of groups (b), network clustering errors (c), and estimated group size (d).

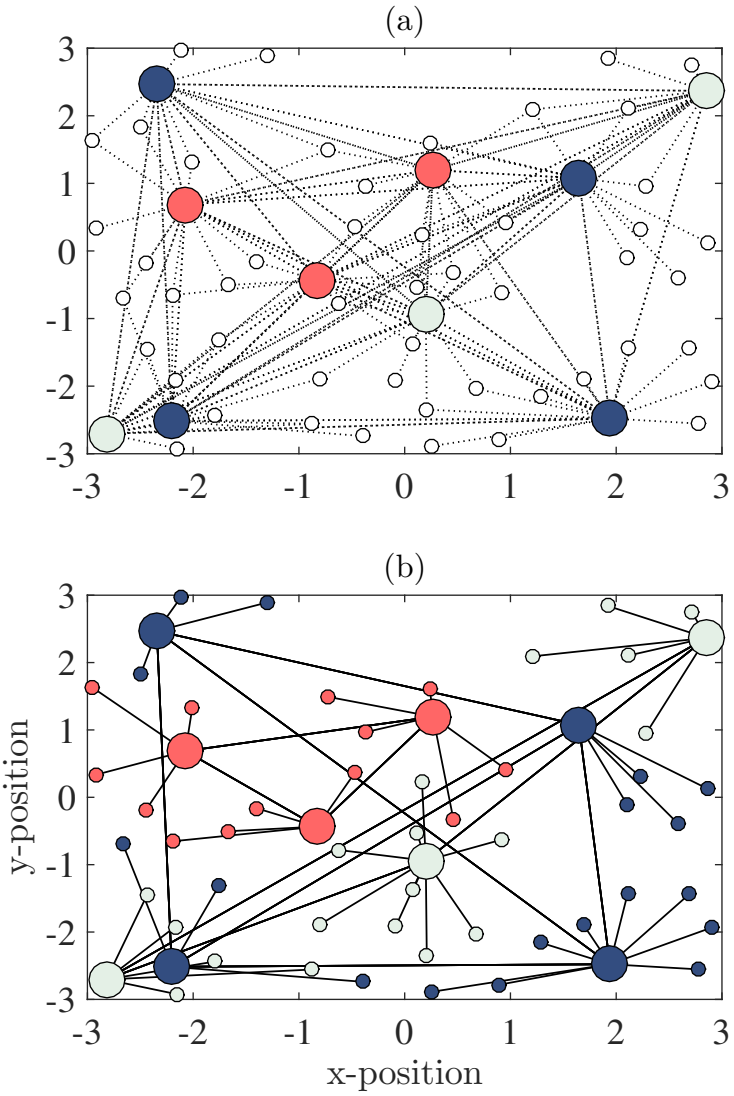

Figure 4.7. Network topology before (a) and after group formation (b), $N = 60$.

Figure 4.6 shows results for the case in which the model assignments change at time instant $i = 400$ for a network of size $N = 60$. The corresponding network topology is given in Fig. 4.7.

This simulation demonstrates the algorithm's ability to track drifts in the models. Some groups may have greater group size than the others because the remainder after the division operation does not equal to zero, i.e., remainder$\{\frac{N^{\mathrm{r}}}{C}\} \neq 0$.

4.4.2 Mobile Network

We consider a network with 40 randomly distributed mobile agents [42]. Agents having the same color belong to the same group. The first 10 agents are informed and denoted by triangles. The uninformed ones are denoted by circles as shown in Fig. 4.9. The informed agents observe data originating from three different models (sources) $C = 3$, where $w_{r_m} \in [100, -100]$. The models are represented by squares. The agents wish to estimate and track these sources in groups. Every agent k updates its location vector $x_{k,i}$ according to the rule:

$$x_{k,i+1} = x_{k,i} + \triangle t \cdot v_{k,i+1} \tag{4.28}$$

where $\triangle t = 0.5$ is a positive time step and $v_{k,i+1}$ is the updated velocity vector of agent k given by

$$v_{k,i+1} = \beta \cdot v^a_{k,i+1} + \gamma v^b_{k,i+1}, \tag{4.29}$$

where β and γ are non-negative weighting factors satisfying $\beta + \gamma = 1$. The velocity vector $v^a_{k,i+1}$, which allows agent k to move towards the desired model is given by

$$v^a_{k,i+1} = \begin{cases} \boldsymbol{w}_{k,i} - x_{k,i}, & \text{if } ||\boldsymbol{w}_{k,i} - x_{k,i}|| \leq \delta, \\ \delta \cdot \frac{\boldsymbol{w}_{k,i} - x_{k,i}}{||\boldsymbol{w}_{k,i} - x_{k,i}||}, & \text{otherwise} \end{cases} \tag{4.30}$$

where we use $\delta = 1$ to bound the agent's speed. The agents should keep a safe distance, $\xi = 3$, from their neighbors to avoid collision during the movement. The velocity vector $v^b_{k,i+1}$ of agent k is given by

$$v^b_{k,i+1} = \frac{1}{|\mathcal{N}^-_{k,i} \cap \ \mathcal{G}_{c,i}|} \sum_{\ell \in \mathcal{N}^-_{k,i} \cap \ \mathcal{G}_{c,i}} \left(1 - \frac{\xi}{||x_{\ell,i} - x_{k,i}||}\right) (x_{\ell,i} - x_{k,i}) \tag{4.31}$$

where $w^\circ_{k,i} = w^\circ_{\mathcal{C}_c}$, $\mathcal{N}^-_{k,i}$ is the neighborhood of agent k excluding itself. We choose $\{\alpha, \beta, \gamma, r_{\mathrm{nei}}\} = \{1, 0.9, 0.1, 20\}$.

Figure 4.8 represents the simulation results which are obtained by averaging over 1000 independent Monte Carlo runs. Figure 4.9 shows the maneuver of the agents with three sources over time. The groups reach these sources supported by equal size of uninformed agents.

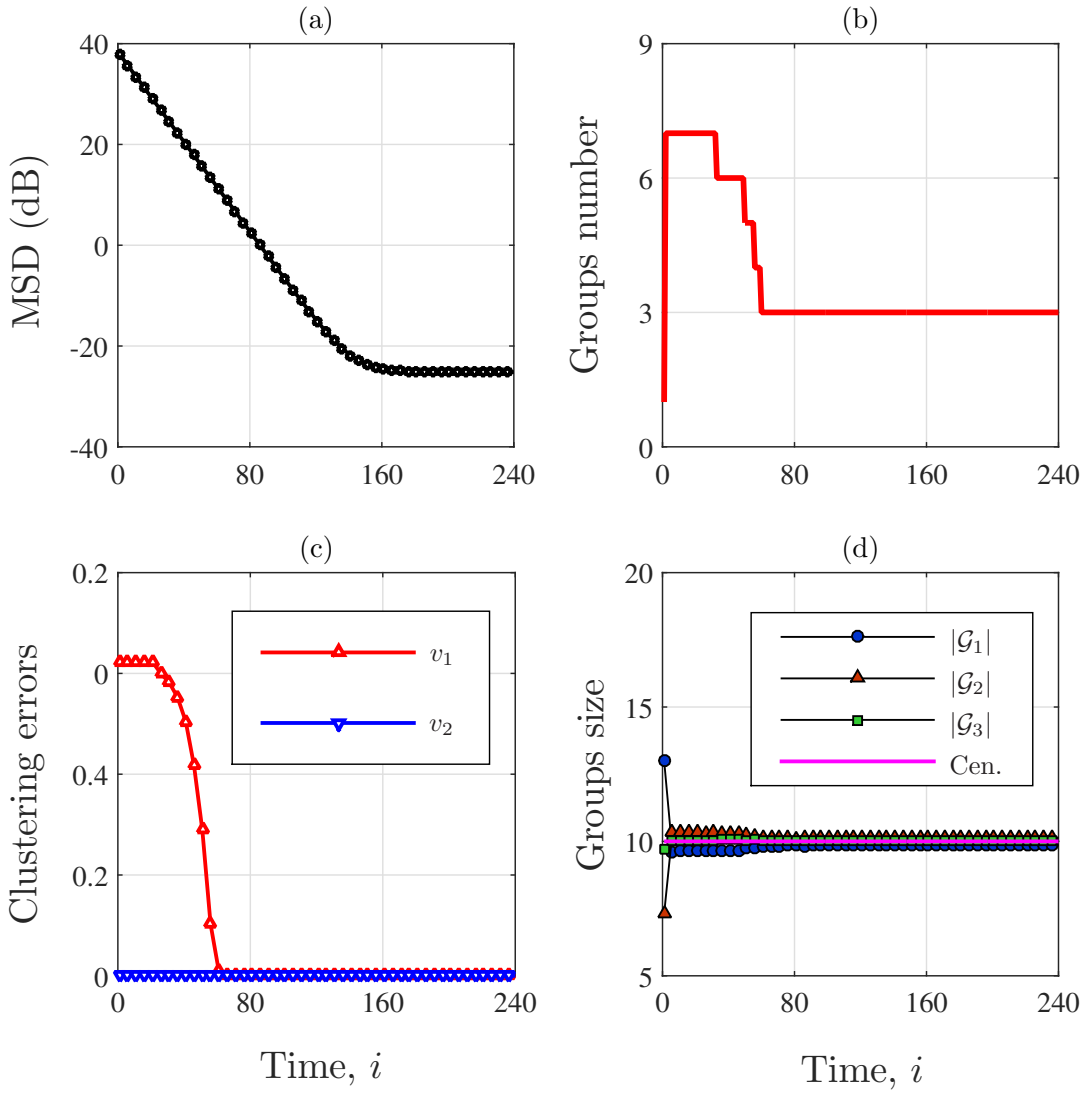

Figure 4.8. Network MSD (a), estimated number of groups (b), network clustering errors (c), and estimated group size (d).

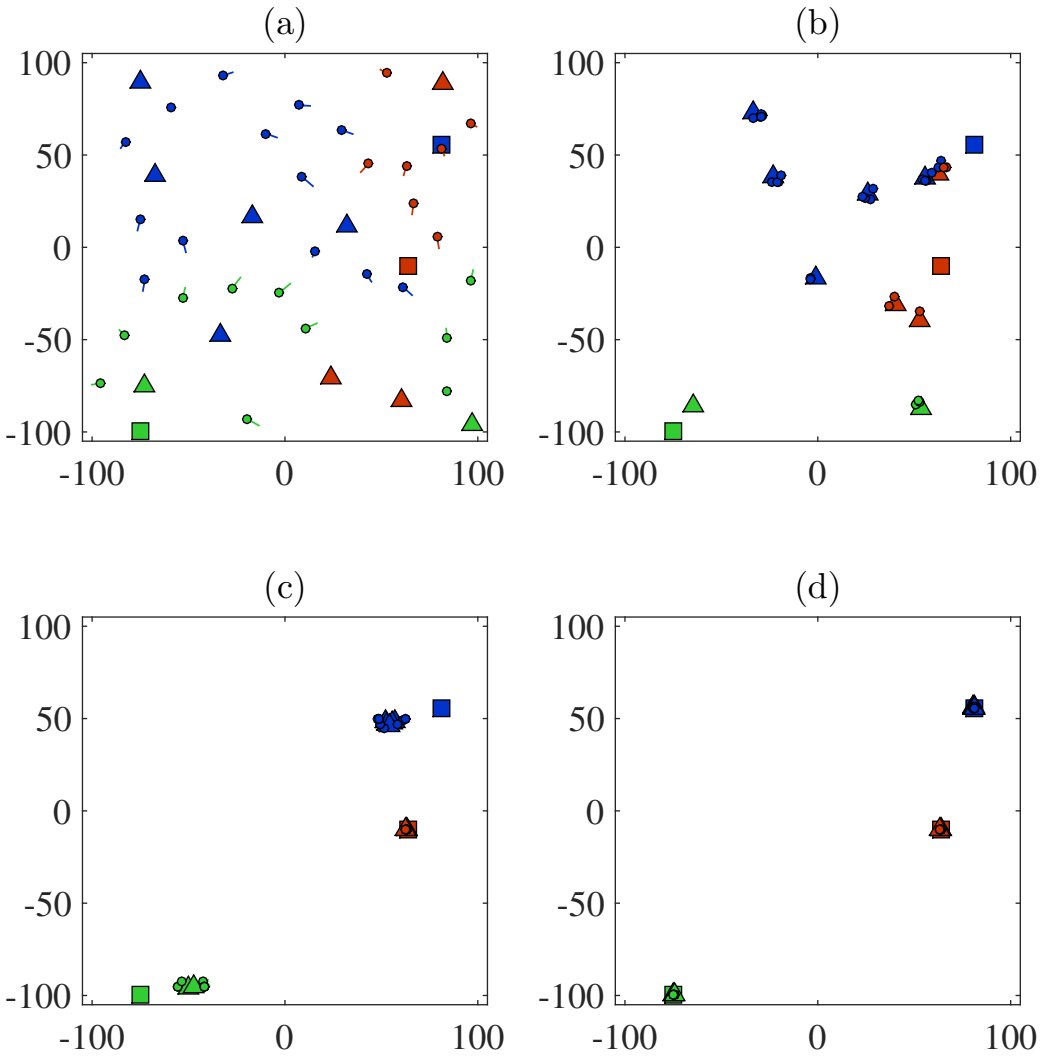

Figure 4.9. Maneuver of the agents with three sources in time instants i=1 (a), i=50 (b), i=100 (c), and i=150 (d). The length unit of the x- and y-axis is the body length of the agents.

Chapter 5

Decentralized Decision-Making Over Adaptive Networks

'If you want to go fast, go alone. If you want to go far, go together.'

African Proverb

In multi-task networks agents are interested in different objectives and do not know beforehand which models are being observed by their neighbors. We have proposed in Chapter 3 a distributed clustering technique that allows the agents to learn and form their clusters from streaming data in a robust manner. In some situations, the agents need to decide between multiple options, e.g., to track only one of multiple sources.

In the works [62, 70], the agents are subject to data arising from two different models and the aim of the network is to reach an agreement to track only one of these two observed sources in a distributed manner. Applying [62] in a mobile environment leads to some undesired scenarios, e.g., splitting the network into two sub-networks before reaching an agreement. The performance of this scheme depends on the initial location of the network and the location of the models. In Sec. 5.1 we replace the proposed classification scheme in [62] by the clustering algorithm proposed in Chapter 3 to ensure a fast and accurate clustering. Moreover, changes in topology over time involve that the network may be separated into two groups before reaching the agreement on one model. Consequently, the decision-making process fails to ensure that the network converges to only one desired model. We add a new term to the velocity control to keep the network moving in a cohesive manner.

In addition, the scheme of the algorithm proposed in [62] is based on binary labeling. This implies that the algorithm can be applied for *only* two different observed models. In Sec. 5.2, we propose a distributed decision-making approach for more than two models, where agents are subject to data arising from more than two different models. The agents need to decide which model to estimate and track. Once the network reaches an agreement on one desired model, the cooperation among the agents enhances the performance of the estimation task by relaying data over the network.

5.1 Decentralized Decision-Making Over Mobile Adaptive Networks

5.1.1 Introduction

Inspired by distinctive biological phenomena, several algorithms are designed to mimic the behavior of animal groups that move together in an amazing coherence, such as bee swarms, birds flying in formation, and schools of fish [42, 45–48]. The agents in some networks need to decide between multiple options, for example, to track only one of two food sources [62].

We consider a distributed mean-square-error estimation problem over an N-agent network. The connectivity of the agents is described by a graph (see Fig. 5.4). Data sensed by any particular agent can arise from one of two different models. The objective is to reach an agreement among all agents in the network on one model to estimate and track. Two definitions are introduced: the observed model, which refers to the one, from which an agent collects data, and the desired model, which refers to the one the agent decides to move towards. The agents do not know which model generated the data they collect; they also do not know which other agents in their neighborhood sense data arising from the same model. Therefore, each agent needs to determine the subset of its neighbors that observes the same model[1].

The desired scenario is shown in Fig. 5.1, where all agents converge to only one model. The proposed classification scheme in [62], which determines the subset neighbors that observes the same model, has some demerits. The performance of this scheme depends on the initial location of the network and the location of the models. These demerits lead to some undesired scenarios, such as, converging to some non-existing model in the middle between the models (see Fig. 5.2).

Since the decision-making objective depends on the classification output, errors made in the classification process have an impact on the global decision. Several clustering algorithms have been proposed in [18, 66, 67]. A fast clustering technique that lets the agents distinguish the neighbors in real-time is needed in mobile networks because the topology changes quickly due to the movement of the agents. In this section, we replace the proposed classification scheme in [62] by the clustering algorithm in Sec. 3.3. Changes in topology over time imply that the network may be separated into two groups

[1]This section is based on the conference paper: S. Khawatmi, X. Huang, and A. M. Zoubir, "Distributed decision-making over mobile adaptive networks," *in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (New Orleans, USA), March 2017, pp. 3864–3868.

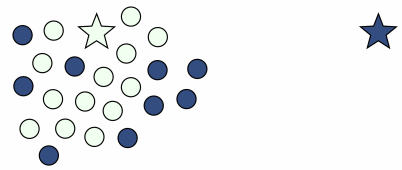

Figure 5.1. Network reaches an agreement among all agents and tracks only one source, where the observed models are represented by two colors. Sources are represented by stars.

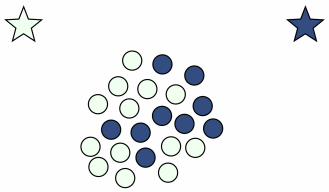

Figure 5.2. Network does not reach an agreement among agents and tracks some non-existing source in the middle between the sources.

Figure 5.3. Network does not reach an agreement among agents and splits into two groups to track the sources.

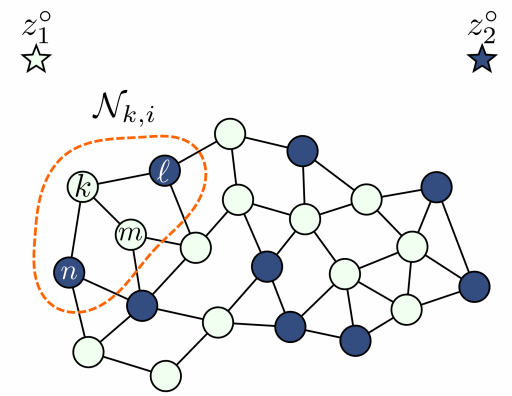

Figure 5.4. Illustration of the network model with two observed models represented by two colors.

before reaching agreement on one model (see Fig. 5.3). Now, while groups are moving far away from each other towards different desired models, they will lose the connections between each other. This means that the decision-making process fails to ensure that the network converges to only one desired model. We add a new term to the velocity control, this term helps to keep the network moving in a cohesive manner, even if agents move and do not make a decision yet.

We present the network and data model in Sec. 5.1.2. Then, we illustrate the decision-making algorithm and the motion mechanism technique in Secs. 5.1.3 and 5.1.4 , respectively. Finally, we demonstrate the method by simulations in Sec. 5.1.5.

5.1.2 Network and Data Model

Consider a collection of N agents distributed in space. Figure 5.4 shows the network structure where agents with the same color observe the same model. The unknown models are denoted by $\{z_1^\circ, z_2^\circ\}$ each of size $M \times 1$. We denote the set of neighbors of agent k at time instant i by $\mathcal{N}_{k,i}$ of size $n_{k,i}$ (i.e, the number of neighbors of agent k). While the set of neighbors of agent k excluding k itself is denoted by $\mathcal{N}_{k,i}^-$. We represent the network topology at time instant i by means of the $N \times N$ adjacency matrix E_i whose entries $e_{\ell k}(i)$ are defined as follows:

$$e_{\ell k}(i) = \begin{cases} 1, & \ell \in \mathcal{N}_{k,i}, \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

We consider an $N \times N$ combination matrix A_i whose $(\ell, k)^{\text{th}}$ entry contains the combination weight $a_{\ell k}(i)$, i.e., the weight that agent k assigns to data received from agent

ℓ. The entries of the combination matrix A_i are non-negative real-valued, satisfying

$$a_{\ell k}(i) = 0 \;\; \text{for } \ell \notin \mathcal{N}_{k,i}, \qquad \sum_{\ell=1}^{N} a_{\ell k}(i) = 1. \tag{5.2}$$

Furthermore, we define the agents observed model vector by

$$w^{\circ} \triangleq \text{col}\,\{w_1^{\circ}, w_2^{\circ}, \ldots, w_N^{\circ}\}, \quad w^{\circ} \in \mathbb{R}^{MN\times 1}. \tag{5.3}$$

Figure 5.4 shows that agent k collects data from model z_1°, i.e., $w_k^{\circ} = z_1^{\circ}$, while agent ℓ collects data from model z_2° which implies $w_{\ell}^{\circ} = z_2^{\circ}$. We denote the estimate vector of the desired model at time instant i of agent k by $\boldsymbol{w}_{k,i}$. We define $w_i \triangleq \text{col}\,\{w_{1,i}, w_{2,i}, \ldots, w_{N,i}\}$. The objective of the network is to have all $w_{k,i}$ converge to only one model, either z_1° or z_2°. We can write that for each agent $k \in \{1, 2, \ldots, N\}$

$$\boldsymbol{w}_{k,i} \to z_j^{\circ} \quad \text{as} \quad i \to \infty \tag{5.4}$$

where j is either 1 or 2. The agents seek to estimate the vector parameter z_j°, which leads to the situation that agents with $w_k^{\circ} = z_j^{\circ}$ track their own observed model, but others with $w_k^{\circ} \neq z_j^{\circ}$ do not track their own observed model, but track z_j° instead, although they do not have any streaming data from z_j°. The aggregate cost function $J^{\text{glob}}(w)$ is defined as

$$J^{\text{glob}}(w) = \sum_{k=1}^{N} ||w_{k,i} - z_j^{\circ}||^2. \tag{5.5}$$

The location and velocity vectors of agent k at time instant i are denoted by $x_{k,i}$ and $v_{k,i}$, respectively. The modified diffusion strategy in [62] is given by the following steps:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} + \mu(\boldsymbol{q}_{k,i} - \boldsymbol{w}_{k,i-1}) \tag{5.6}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell\in\mathcal{N}_{k,i}} (\dot{\boldsymbol{a}}_{\ell k}(i)\boldsymbol{\psi}_{\ell,i} + \ddot{\boldsymbol{a}}_{\ell k}(i)\boldsymbol{w}_{\ell,i-1}) \tag{5.7}$$

where μ is a positive step-size parameter and $\boldsymbol{q}_{k,i}$ is the noisy location of the model that agent k observes and is given by

$$\boldsymbol{q}_{k,i} = w_k^{\circ} + \boldsymbol{\eta}_{k,i} \tag{5.8}$$

where $\boldsymbol{\eta}_{k,i}$ is a zero-mean white random process with variance $\sigma_k^2(i) = \kappa||w_k^{\circ} - x_{k,i}||^2$, for $\kappa > 0$ (see Figs. 5.5 and 5.6).

The combination coefficients $\{\dot{\boldsymbol{a}}_{\ell k}(i)\}$ and $\{\ddot{\boldsymbol{a}}_{\ell k}(i)\}$ are two sets of non-negative entries in the combination matrices $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$ which satisfy:

$$\dot{\boldsymbol{A}}_i + \ddot{\boldsymbol{A}}_i = A_i. \tag{5.9}$$

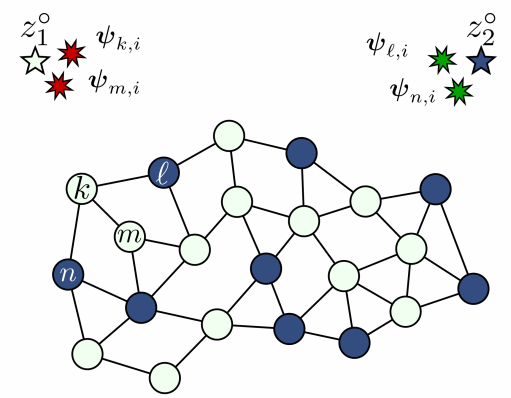

Figure 5.5. Illustration of the intermediate estimate $\boldsymbol{\psi}_i$ for some agents in the network.

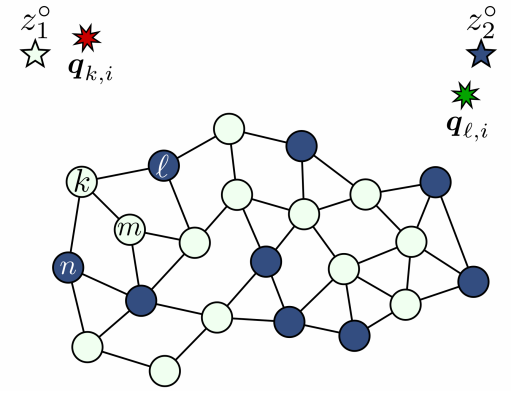

Figure 5.6. Illustration of the noisy location $\boldsymbol{q}_{k,i}$ ($\boldsymbol{q}_{\ell,i}$) of the model that agent k (ℓ) observes.

The design method of the matrices $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$ is detailed in Sec. 5.1.3. We define the central location and velocity of the network at time instant i by averaging location and velocity of all agents over the network, respectively, as

$$x_i^\circ \triangleq \frac{1}{N}\sum_{k=1}^{N} x_{k,i}, \quad v_i^\circ \triangleq \frac{1}{N}\sum_{k=1}^{N} v_{k,i}. \tag{5.10}$$

The central network location x_i° and velocity v_i° are estimated using the diffusion strategy in a distributed manner by considering the following global cost functions:

$$J^x(x^g) = \sum_{k=1}^{N} ||x_{k,i}^g - x_i^\circ||^2, \quad J^v(v^g) = \sum_{k=1}^{N} ||v_{k,i}^g - v_i^\circ||^2. \tag{5.11}$$

Applying the diffusion strategy structure to estimate $\boldsymbol{x}_{k,i}^g$ and $\boldsymbol{v}_{k,i}^g$, respectively, we obtain

$$\boldsymbol{\theta}_{k,i} = \boldsymbol{x}_{k,i-1}^g + \mu(x_{k,i} - \boldsymbol{x}_{k,i-1}^g) \tag{5.12}$$

$$\boldsymbol{x}_{k,i}^g = \sum_{\ell\in\mathcal{N}_{k,i}} a_{\ell k}(i)\boldsymbol{\theta}_{\ell,i} \tag{5.13}$$

$$\boldsymbol{\phi}_{k,i} = \boldsymbol{v}_{k,i-1}^g + \mu(v_{k,i} - \boldsymbol{v}_{k,i-1}^g) \tag{5.14}$$

$$\boldsymbol{v}_{k,i}^g = \sum_{\ell\in\mathcal{N}_{k,i}} a_{\ell k}(i)\boldsymbol{\phi}_{\ell,i}. \tag{5.15}$$

Note that we estimate the central network location and velocity using matrix A_i rather than $\dot{\boldsymbol{A}}_i$ or $\ddot{\boldsymbol{A}}_i$ due to the fact that the agents are required to share velocity and location information with all neighbors, regardless of their observed models.

Let the true clustering matrix, which gives information about the observed model and is not known beforehand, be denoted by F_i°. The assignment of the $(\ell, k)^{\text{th}}$ entry to one means

$$f_{\ell k}^\circ(i) = 1 \Rightarrow \{\ell \in \mathcal{N}_{k,i} \text{ and } w_k^\circ = w_\ell^\circ\}. \tag{5.16}$$

We apply the clustering technique proposed in Sec. 3.3 to create the estimated clustering matrix $\boldsymbol{F}_i$ of size $N\times N$. Each agent k runs the following steps for the clustering process:

$$\boldsymbol{\psi}_{k,i}^c = \boldsymbol{\psi}_{k,i-1}^c + \mu_k(\boldsymbol{q}_{k,i} - \boldsymbol{\psi}_{k,i-1}^c) \tag{5.17}$$

$$\boldsymbol{w}_{k,i}^c = \sum_{\ell\in\boldsymbol{\mathcal{N}}'_{k,i}} \boldsymbol{a}'_{\ell k}(i)\boldsymbol{\psi}_{\ell,i}^c \tag{5.18}$$

$\boldsymbol{\psi}_{k,i}^c$ and $\boldsymbol{w}_{k,i}^c$ both converge to the observed model w_k° without being affected by the decision process. We set $\boldsymbol{\psi}_{k,-1}^c = 0$ and $\boldsymbol{B}_{-1} = \boldsymbol{S}_{-1} = \boldsymbol{F}_{-1} = I_N$. The combination matrix $\boldsymbol{A}'_i$ is designed using the following steps [18]:

$$\boldsymbol{b}_{\ell k}(i) = \begin{cases} 1, & \text{if } ||\boldsymbol{\psi}_{\ell,i}^c - \boldsymbol{w}_{k,i-1}^c|| \le \epsilon, \\ 0, & \text{otherwise} \end{cases} \tag{5.19}$$

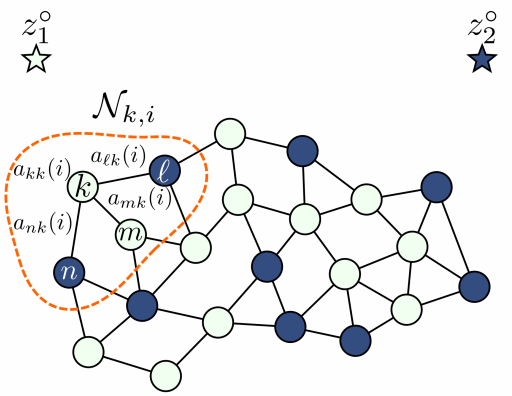

Figure 5.7. Example of agent's k neighbors with the combination weights $\{a_{\ell k}(i)\}$.

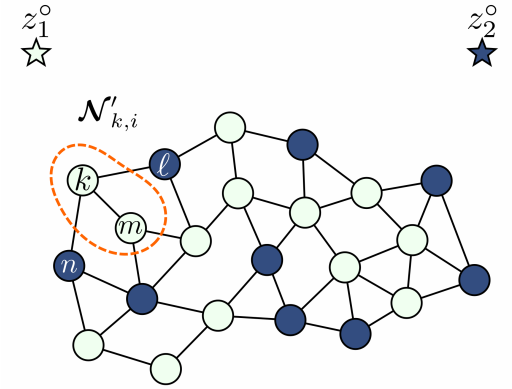

Figure 5.8. Example of the clustered neighbors of agent k that observe the same model.

$$\boldsymbol{s}_{\ell k}(i) = \xi \times \boldsymbol{s}_{\ell k}(i-1) + (1-\xi) \times \boldsymbol{b}_{\ell k}(i) \tag{5.20}$$

$$\boldsymbol{f}_{\ell k}(i) = \lfloor \boldsymbol{s}_{\ell k}(i) \rceil \tag{5.21}$$

for $\epsilon > 0$, $0 \leq \xi \leq 1$, and $0 < \zeta \leq 1$. The combination coefficients $\{\boldsymbol{a}'_{\ell k}(i)\}$ satisfy

$$\boldsymbol{a}'_{\ell k}(i) = 0, \quad \text{if} \quad \ell \notin \boldsymbol{\mathcal{N}}'_{k,i}, \quad \sum_{\ell=1}^{N} \boldsymbol{a}'_{\ell k}(i) = 1 \tag{5.22}$$

where $\boldsymbol{\mathcal{N}}'_{k,i}$ consists of neighbors believing that they belong to the same cluster, i.e., $\boldsymbol{f}_{\ell k}(i) = 1$ implies $\ell \in \boldsymbol{\mathcal{N}}'_{k,i}$ (see Figs. 5.7 and 5.8).

5.1.3 Selection of Combination Matrices

Applying the strategy in [62], we use matrix $\boldsymbol{G}_i$ of size $N \times N$ to estimate the desired model of each agent. Agent k assigns the value of the $(\ell, k)^{\text{th}}$ entry for each agent

$\ell \in \mathcal{N}_{k,i}^-$ according to (5.24). The meaning of the value is as follows:

$$\begin{cases} \boldsymbol{g}_{\ell k}(i) = 1: & w_{\ell,i} \to w_k^\circ, \\ \boldsymbol{g}_{\ell k}(i) = 0: & w_{\ell,i} \not\to w_k^\circ. \end{cases} \tag{5.23}$$

Since agent k has access to the desired models of its neighbors $\boldsymbol{g}_{\ell\ell}(i)$, it adjusts its estimate for desired model of agent ℓ, $\boldsymbol{g}_{\ell k}(i)$, according to the following rule:

$$\boldsymbol{g}_{\ell k}(i) = \begin{cases} \boldsymbol{g}_{\ell\ell}(i-1), & \text{if } \boldsymbol{f}_{\ell k}(i) = 1, \\ 1 - \boldsymbol{g}_{\ell\ell}(i-1), & \text{otherwise.} \end{cases} \tag{5.24}$$

Each diagonal entry $\boldsymbol{g}_{kk}(i)$ indicates whether agent k wishes to track its own observed model or not, i.e.,

$$\begin{cases} \boldsymbol{g}_{kk}(i) = 1: & \boldsymbol{w}_{k,i} \to w_k^\circ, \\ \boldsymbol{g}_{kk}(i) = 0: & \boldsymbol{w}_{k,i} \not\to w_k^\circ \end{cases} \tag{5.25}$$

where agent k updates its desired model $\boldsymbol{g}_{kk}(i)$ according to:

$$\boldsymbol{g}_{kk}(i) = \begin{cases} \boldsymbol{g}_{kk}(i-1), & \text{with probability } \boldsymbol{p}_k(i), \\ 1 - \boldsymbol{g}_{kk}(i-1), & \text{with probability } 1 - \boldsymbol{p}_k(i) \end{cases} \tag{5.26}$$

and $\boldsymbol{p}_k(i)$ is given by

$$\boldsymbol{p}_k(i) = \frac{[\boldsymbol{n}_k^g(i)]^K}{[\boldsymbol{n}_k^g(i)]^K + [n_k(i) - \boldsymbol{n}_k^g(i)]^K} \tag{5.27}$$

for a positive constant K. Figure 5.9 shows examples for the probability $\boldsymbol{p}_k(i)$ using different values of K. $\boldsymbol{n}_k^g(i)$ is the size of the set $\boldsymbol{\mathcal{N}}_{k,i}^g$ that contains the subset of agents that are in the neighborhood of agent k and have the same desired model as agent k at time instant $i-1$. That is, $\boldsymbol{\mathcal{N}}_{k,i}^g$ is constructed as follows:

$$\boldsymbol{\mathcal{N}}_{k,i}^g = \{\ell | \ell \in \mathcal{N}_{k,i},\ \boldsymbol{g}_{\ell k}(i) = \boldsymbol{g}_{kk}(i-1)\}. \tag{5.28}$$

The entries of $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$ are set according to the following rules:

$$\dot{\boldsymbol{a}}_{\ell k}(i) = \begin{cases} a_{\ell k}(i), & \text{if } \ell \in \mathcal{N}_{k,i} \text{ and } \boldsymbol{f}_{\ell k}(i) = \boldsymbol{g}_{kk}(i), \\ 0, & \text{otherwise.} \end{cases} \tag{5.29}$$

$$\ddot{\boldsymbol{a}}_{\ell k}(i) = \begin{cases} a_{\ell k}(i), & \text{if } \ell \in \mathcal{N}_{k,i} \text{ and } \boldsymbol{f}_{\ell k}(i) \neq \boldsymbol{g}_{kk}(i), \\ 0, & \text{otherwise.} \end{cases} \tag{5.30}$$

As a result, in Eq. (5.7) agent k combines $\boldsymbol{\psi}_{\ell,i}$ if it wishes to estimate w_ℓ°, otherwise it combines $\boldsymbol{w}_{\ell,i-1}$ instead, where $\ell \in \mathcal{N}_{k,i}$, (see Figs. 5.10 and 5.11).

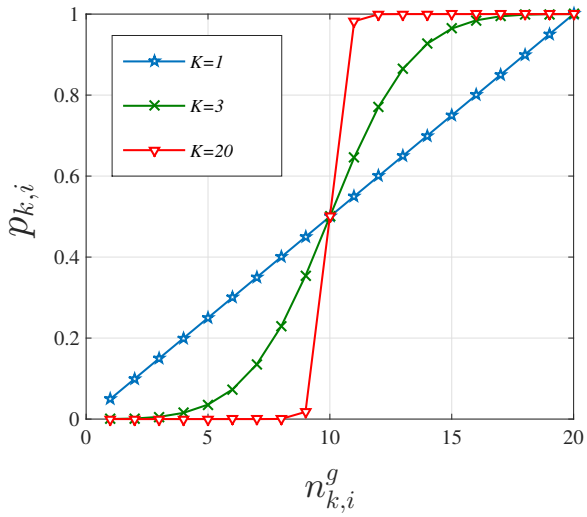

Figure 5.9. Probability $\boldsymbol{p}_k(i)$ for different values of K.

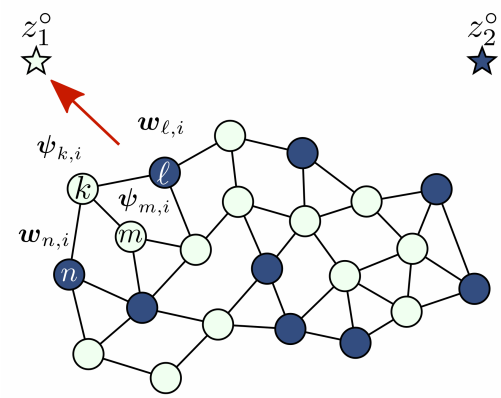

Figure 5.10. Data types that agent k combines from its neighbors in case that the desired model of agent k is z_1°.

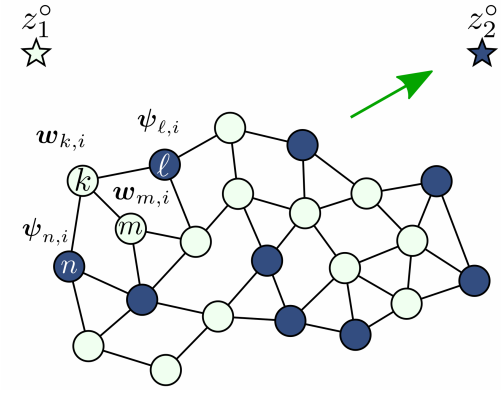

Figure 5.11. Data types that agent k combines from its neighbors in case that the desired model of agent k is z_2°.

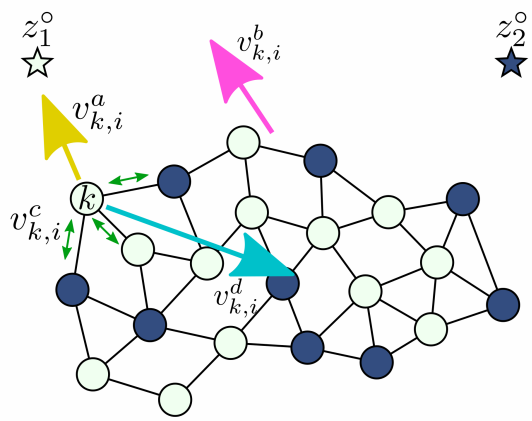

Figure 5.12. Illustration of the main terms of the velocity equation.

5.1.4 Motion Model

Every agent k updates its location vector according to the rule

$$x_{k,i+1} = x_{k,i} + \triangle t \cdot v_{k,i+1} \tag{5.31}$$

where $\triangle t$ is a positive time step and $v_{k,i+1}$ is the updated velocity vector of agent k. Several factors influence the determination of the velocity $v_{k,i+1}$ of agent k, such as: the desire to move towards the desired model z_j°, the desire to move in coordination with other agents, and the desire to avoid collision. Figure 5.12 shows four velocity factors. The velocity vector $v_{k,i+1}^a$, which allows agent k to move towards the desired model, is given by

$$v_{k,i+1}^a = \begin{cases} \boldsymbol{w}_{k,i} - x_{k,i} \ , & \text{if } ||\boldsymbol{w}_{k,i} - x_{k,i}|| \leq \delta, \\ \delta \cdot \frac{\boldsymbol{w}_{k,i} - x_{k,i}}{||\boldsymbol{w}_{k,i} - x_{k,i}||} \ , & \text{otherwise} \end{cases} \tag{5.32}$$

where δ is a positive scaling factor used to bound the agent's speed. To move in a harmonious manner, the velocity vector $v_{k,i+1}^b$ of agent k is updated as, $v_{k,i+1}^b = \boldsymbol{v}_{k,i}^g$. Agents should keep a safe distance r from their neighbors to avoid collisions during the movement. The velocity vector $v_{k,i+1}^c$ of agent k is given by

$$v_{k,i+1}^c = \frac{1}{n_k(i) - 1} \sum_{\ell \in \mathcal{N}_{k,i}^-} \left(1 - \frac{r}{||x_{\ell,i} - x_{k,i}||}\right) (x_{\ell,i} - x_{k,i}). \tag{5.33}$$

Before reaching agreement on one desired model, the network might become separated into two groups, each group moving towards its desired model. If these two groups move away from each other and lose connections, the decision-making process fails. To resolve this issue, we define an $N \times 1$ vector $\boldsymbol{\iota}_i$, each entry $\boldsymbol{\iota}_k(i)$ is given by

$$\boldsymbol{\iota}_k(i) = \begin{cases} 1, & \text{if } \boldsymbol{p}_k(i) > 0.5 \text{ AND } \ \boldsymbol{n}_k^g(i) < n_k(i), \\ 0, & \text{otherwise.} \end{cases} \tag{5.34}$$

The condition in Eq. (5.34) implies that $\mathcal{N}_{k,i}$ does not agree yet on one desired model and $\boldsymbol{p}_k(i) > 0.5$ (i.e., agent k will keep its previous decision). If $\boldsymbol{\iota}_k(i) = 1$, an additional action has to be taken by agent k, where $\boldsymbol{\iota}_k(i)$ controls the term $v^d_{k,i+1}$ that enforces agent k to move towards the center of the network in order to keep the network cohesive. Herein, $v^d_{k,i+1}$ is given by

$$v^d_{k,i+1} = \frac{\boldsymbol{x}^g_{k,i} - x_{k,i}}{||\boldsymbol{x}^g_{k,i} - x_{k,i}||}. \tag{5.35}$$

Finally, for the non-negative weighting factors λ and β satisfying: $\lambda + \beta = 1$, each agent adjusts its velocity according to the following rule:

$$\begin{aligned} v_{k,i+1} =& \big[1 - \boldsymbol{\iota}_k(i)\big] \cdot \big[\lambda \cdot v^a_{k,i+1} + \beta v^b_{k,i+1}\big] \\ &+ \boldsymbol{\iota}_k(i) v^d_{k,i+1} + v^c_{k,i+1}. \end{aligned} \tag{5.36}$$

5.1.5 Simulation Results

We consider a connected network with 40 randomly distributed agents. The maximum number of neighbors of every agent k is $n_{k,i} = 7$, as long as they are within radius $R = 15$. Agents observe data originating from two different models: $z^\circ_1 = [-10; 10]$ and $z^\circ_2 = [10; 10]$. The assignment of agents to the models is random. We use a uniform combination policy to generate the coefficients $\{a_{\ell k}(i)\}$ and $\{a'_{\ell k}(i)\}$. The clustering parameters are set as follows: $\{\epsilon, \xi\} = \{5, 0.6\}$. The velocity parameters are set as follows: $\{\lambda, \beta, r, \Delta t, \delta\} = \{0.2, 0.8, 3, 0.1, 1\}$ and $\{\mu, \kappa, K\} = \{0.05, 0.02, 20\}$. The simulation results are obtained by averaging over 1000 independent experiments with different setup of the initial network location.

Table 5.2 displays the success rate R_r of the decision-making to agree on one model and the average time required to achieve this agreement T_r. Obviously, the proposed strategy provides better performance with almost 100% success rate while on average needing compared to [62] fewer iterations to achieve agreement.

	T_r(sec)	R_r(%)
Strategy [62]	72	64.2%
Proposed strategy	58	99.3%

Table 5.1. Decision-making success rate R_r and the average time required to achieve agreement T_r .

The transient network mean-square deviation (MSD) at each time instant i is defined by

$$\text{MSD}_d(i) \triangleq \frac{1}{N} \sum_{k=1}^{N} ||z^\circ_d - \boldsymbol{w}_{k,i}||^2 \tag{5.37}$$

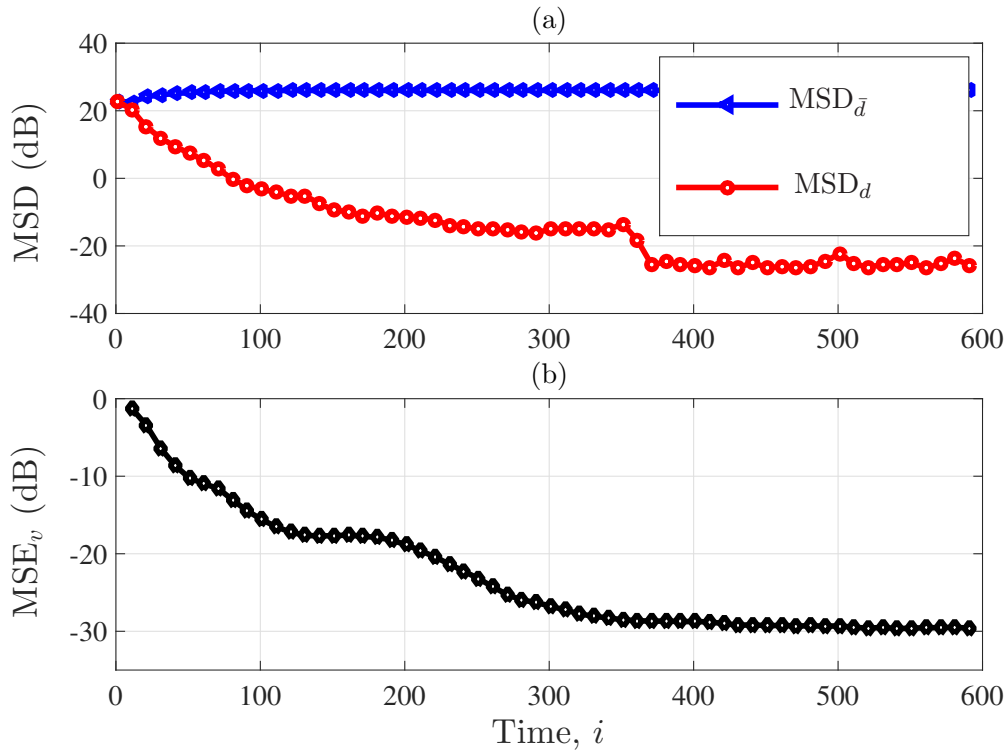

Figure 5.13. Transient network mean-square deviation MSD (a). Transient network mean-square error MSE$_v$ (b).

where z_d° is the desired model. Figure 5.13(a) depicts two curves and shows how the network converges to z_d°. By substituting the undesired model $z_{\bar{d}}^\circ$ in (5.37), we obtain the second curve MSD$_{\bar{d}}$ that is shown in Fig. 5.13(a). Figure 5.13(b) shows the transient network mean-square-error of estimating the central velocity v_i° which is given by

$$\text{MSE}_v(i) \triangleq \frac{1}{N}\sum_{k=1}^{N} ||v_i^\circ - \boldsymbol{v}_{k,i}^g||^2. \tag{5.38}$$

The learning curve indicates that the network moves coherently.

Figure 5.14 depicts the maneuver of fish schools with two food sources over time where the agents agree on the model z_1°.

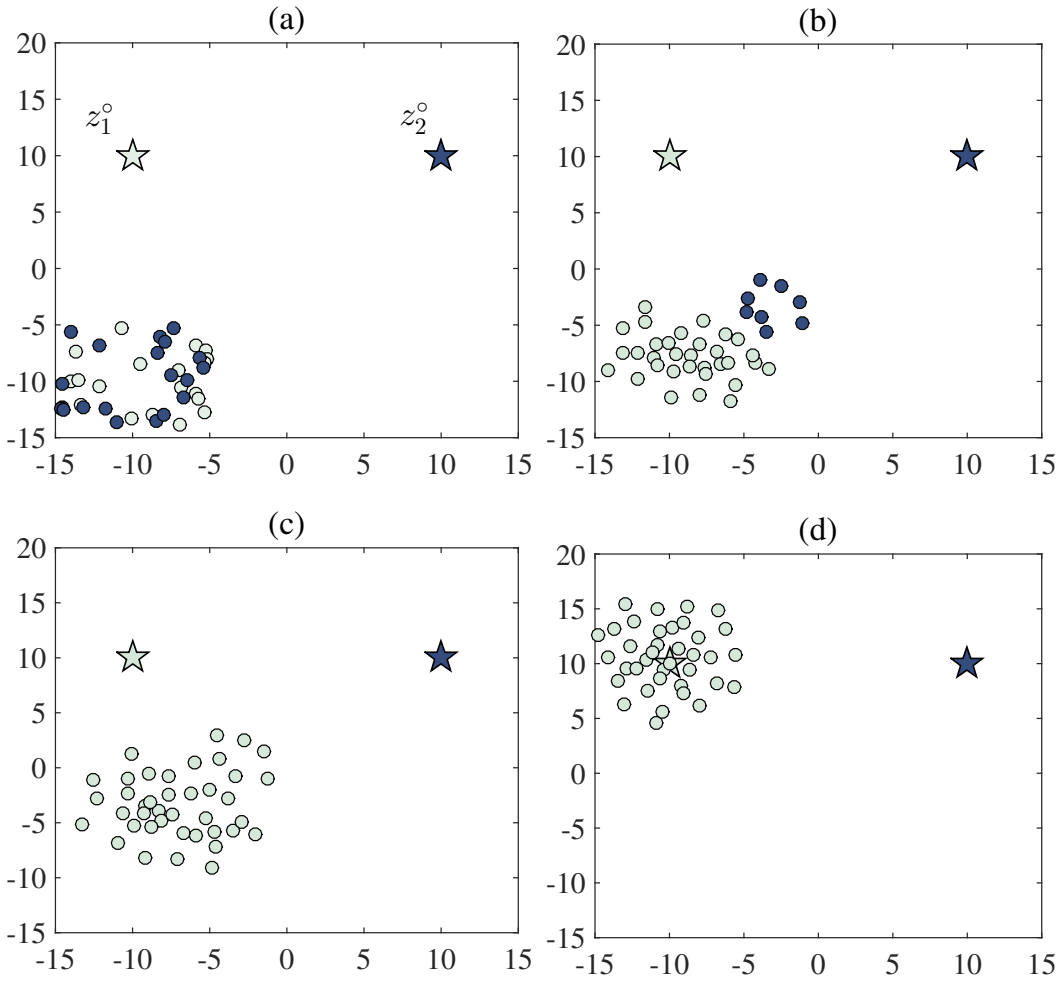

Figure 5.14. Maneuver of fish schools with two food sources in time instants $i = 10$ (a), $i = 30$ (b), $i = 100$ (c), and $i = 500$ (d). The length unit of the x- and y-axis is the body length of the agents.

5.2 Decentralized Decision-Making Over Multi-Task Networks

5.2.1 Introduction

In many multi-task networks that have multiple objectives, agents need to decide between these multiple objectives and reach an agreement to have only one objective of the network. In the works [21, 62] agents in the network are subject to data arising from two different models and the aim of the network is to reach agreement to track only one of these two observed sources in a distributed manner. The strategy of the algorithm proposed in [62] is based on binary labeling which implies that the algorithm can only be applied for two different observed models. In this section, we propose an approach for scenarios, where agents are subject to data arising from more than two different models[2].

We consider a distributed mean-square-error estimation problem over an N-agent network. The connectivity of the agents is described by a graph (see Fig. 5.15). Data sensed by any particular agent can arise from one of different models. The objective is to reach an agreement among all agents in the network on one common model to estimate. We recall the following definitions: the observed model, which refers to the one from which an agent collects data, and the desired model, which refers to the one the agent decides to estimate. The agents do not know which model generated the data they collect; they also do not know which other agents in their neighborhood sense data arising from the same model. Therefore, each agent needs to determine the subset of its neighbors that observes the same model. The distinguishing process of the neighbors is called the clustering process. Since the decision-making objective depends on the clustering output, errors made during the clustering process have an impact on the global decision. We apply the clustering scheme proposed in [18] to ensure an accurate clustering output.

We describe the network and data model in Sec. 5.2.2. Then, we illustrate the local labeling system and the decision-making algorithm with the performance analysis of the algorithm in Secs. 5.2.3, 5.2.4, and 5.2.5, respectively. In Sec. 5.2.6 we extend the algorithm to a special case when the whole network follows the model of a specific agent. Finally, simulation results and discussions are presented in Sec. 5.2.7.

[2]This section is based on the journal paper: S. Khawatmi, A. H. Sayed, and A. M. Zoubir, "Decentralized decision-making over multi-task networks," (under review), 2017.

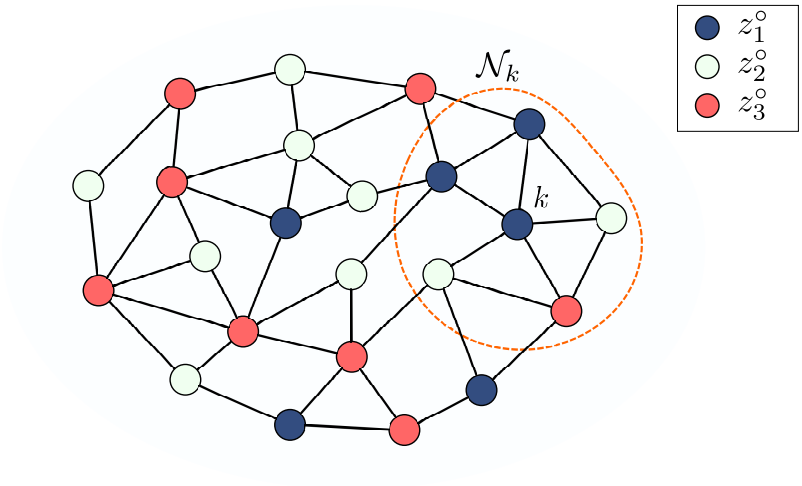

Figure 5.15. Example of a network topology, agents with the same color observe the same model.

5.2.2 Network and Data Model

Consider a collection of N agents distributed in space, as illustrated in Fig. 5.15. We represent the network topology by means of an $N \times N$ adjacency matrix E whose entries $e_{\ell k}$ are defined as follows:

$$e_{\ell k} = \begin{cases} 1, & \ell \in \mathcal{N}_k, \\ 0, & \text{otherwise} \end{cases} \tag{5.39}$$

where $\mathcal{N}_k$ is the set of neighbors of agent k (we denote its size by n_k). We also write $\mathcal{N}_k^-$ to denote the same neighborhood excluding agent k.

Figure 5.15 shows the network structure where agents with the same color observe the same model. We denote the unknown models by $\{z_1^\circ, \ldots, z_C^\circ\}$, each of size $M \times 1$ where $C \leq N$. Each agent k observes data generated by one of these C unknown models. We denote the model observed by agent k by w_k°. Figure 5.15 shows that agent k collects data from model z_1°, in which case $w_k^\circ = z_1^\circ$. For any other agent ℓ observing the same model z_1°, it will hold that $w_\ell^\circ = z_1^\circ$. We stack the $\{w_k^\circ\}$ into a column vector:

$$w^\circ \triangleq \text{col}\,\{w_1^\circ, w_2^\circ, \cdots, w_N^\circ\}, \quad w^\circ \in \mathbb{R}^{MN \times 1}. \tag{5.40}$$

At every time instant i, every agent k has access to a scalar measurement $\boldsymbol{d}_k(i)$ and a $1 \times M$ regression vector $\boldsymbol{u}_{k,i}$. The measurements across all agents are assumed to be generated via linear regression models of the form:

$$\boldsymbol{d}_k(i) = \boldsymbol{u}_{k,i} w_k^\circ + \boldsymbol{v}_k(i). \tag{5.41}$$

All random processes are assumed to be stationary. Moreover, $\boldsymbol{v}_k(i)$ is a zero-mean white measurement noise that is independent over space and has variance $\sigma_{v,k}^2$. The

regression data $\boldsymbol{u}_{k,i}$ is assumed to be a zero-mean Gaussian process, independent over time and space, and independent of $\boldsymbol{v}_\ell(j)$ for all k, ℓ, i, j. We denote the covariance matrix of $\boldsymbol{u}_{k,i}$ by $R_{u,k} = \mathbb{E}\, \boldsymbol{u}_{k,i}^{\mathsf{T}} \boldsymbol{u}_{k,i}$.

Agents do not know which model is generating their data. They also do not know which models are generating the data of their neighbors. Still, we need to apply a learning strategy that will allow the agents to converge to their individual driving models, while also learning which of their neighbors share the same model. Using the algorithm proposed in [18], each agent k repeats the following steps involving an LMS adaptation update followed by an aggregation step:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{\psi}_{k,i-1}) \tag{5.42}$$

$$\boldsymbol{\phi}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) \boldsymbol{\psi}_{\ell,i} \tag{5.43}$$

where μ_k is the step-size used by agent k. The intermediate estimate vector and the estimate vector of the observed model by agent k at time instant i are denoted by $\boldsymbol{\psi}_{k,i}$ and $\boldsymbol{\phi}_{k,i}$, respectively. We collect the estimated vectors across all agents into the aggregate vector:

$$\boldsymbol{\phi}_i \triangleq \text{col}\, \{\boldsymbol{\phi}_{1,i}, \boldsymbol{\phi}_{2,i}, \cdots, \boldsymbol{\phi}_{N,i}\}. \tag{5.44}$$

In a manner similar to [18], we introduce a clustering matrix $\boldsymbol{E}_i$. Its structure is similar to the adjacency matrix E, with ones and zeros, except that the value at location (ℓ, k) will be set to one if agent k *believes* at time instant i that its neighbor ℓ belongs to the same cluster, i.e., observes the same model

$$\boldsymbol{e}_{\ell k}(i) = \begin{cases} 1, & \text{if } \ell \in \mathcal{N}_k \text{ and } k \text{ believes that } w_k^\circ = w_\ell^\circ, \\ 0, & \text{otherwise.} \end{cases} \tag{5.45}$$

These entries help define the neighborhood set $\boldsymbol{\mathcal{N}}_{k,i}$, which consists of all neighbors at time instant i that agent k believes share the same model. The entries $\{\boldsymbol{a}_{\ell k}(i)\}$ in (5.43) are non-negative scalars that satisfy

$$\boldsymbol{a}_{\ell k}(i) = 0 \;\; \text{for } \ell \notin \boldsymbol{\mathcal{N}}_{k,i}, \qquad \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) = 1. \tag{5.46}$$

Although there is a multitude of models generating the data that is feeding into the agents, namely, $\{z_1^\circ, z_2^\circ, \ldots, z_C^\circ\}$, the objective is to develop a strategy that will allow all agents to converge towards one of these models. We refer to this particular choice as the desired model and denote it by z_d°.

In this way, an agent whose source (observed) model agrees with the desired model, i.e., $w_k^\circ = z_d^\circ$, it will end up tracking its own source. On the other hand, an agent whose

source model is not the desired model, i.e., $w_k^\circ \neq z_d^\circ$, it will track z_d° instead although it is sensing data generated by a different model. We define the estimate vector of agent's k desired model by $\boldsymbol{w}_{k,i}$. The reason behind indicating $\boldsymbol{w}_{k,i}$ as the estimate vector of *agent's* k desired model instead of the *network's* desired model is that the agents may have different desired models before the convergence (steady-state). Once the agents reach the agreement among each other on only one model, we can refer to $\boldsymbol{w}_{k,i}$ as the estimate vector by agent k of the *network's* desired model. For the initialization at time instant $i = 1$, each agent assigns $\boldsymbol{w}_{k,0} = \boldsymbol{\psi}_{k,1}$ (i.e. at time instant $i = 1$ the desired model of each agent is its own source model). The decision-making process drives the desired models of all agents to converge to only one model. For example, if the agents observe $C = 5$ different models, the number of the desired models in the network will decrease with iterations gradually form 5 models till only one model. This is achieved by switching the estimate $\boldsymbol{w}_{k,i}$ of some agents during the decision-making process according to some conditions that are explained later. However, agents do not know which models are desired by their neighbors at each time instant i. Thus, we need to develop a learning strategy that allows the agents to distinguish the individual desired models of their neighbors.

It turns out that, in order for the objective of the network to be met, it is important for agents to combine the estimates of their neighbors in a judicious manner because, unbeknown to the agents some of their neighbors may be wishing to estimate different models. If cooperation is performed blindly with all neighbors, then performance can deteriorate with agents converging to non-existing locations. For this reason, and motivated by the discussion from [62], we add the step (5.47) below after (5.42) and (5.43), which involves two sets of combination coefficients from two matrices $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$. There are two main ideas behind the construction (5.47). First, is to let each agent k cooperate only with the subset of the neighbors that share the same desired model as it does. Second, is to let each agent k combine $\boldsymbol{\phi}_{\ell,i}$ if the desired model of agent k at time instant i is the same as ℓ's observed model. Otherwise, agent k combines the previous value of $\boldsymbol{w}_{\ell,i-1}$ from agent ℓ that has the same desired model at time instant i as agent k:

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \dot{\boldsymbol{a}}_{\ell k}(i)\boldsymbol{\phi}_{\ell,i} + \sum_{\ell=1}^{N} \ddot{\boldsymbol{a}}_{\ell k}(i)\boldsymbol{w}_{\ell,i-1}. \tag{5.47}$$

Note that the matrices $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$ are not constructed from matrix $\boldsymbol{A}_i$. The selection of the non-negative coefficients $\{\dot{\boldsymbol{a}}_{\ell k}(i)\}$ and $\{\ddot{\boldsymbol{a}}_{\ell k}(i)\}$ is explained in Sec. 5.2.4.

We summarize the main five steps of the approach:

1. Learning the observed models of the neighbors. This step is performed by building

the matrix $\boldsymbol{E}_i$ in step (5.45). The information provided by each entry $\boldsymbol{e}_{\ell k}(i)$ is whether the corresponding agents ℓ and k have the same observed model or not.

2. Learning and labeling the desired model of the neighbors at each time instant i. This step allows the agents to distinguish the individual desired models of their neighbors at time instant i. The information provided by this step is the number of different models that are desired by neighbors and how many each model is repeated at time instant i among neighbors.

3. Decision-making step by switching the desired model of some agents to let the network converge to only one model.

4. Learning the desired models of the neighbors after the switching step. This step is performed by building the matrix $\boldsymbol{H}_i$ in step (5.54) in Sec. 5.2.4. The information provided by each entry $\boldsymbol{h}_{\ell k}(i)$ is whether the corresponding agents ℓ and k have the same desired model or not after the switching step.

5. Updating the estimate vectors $\{\boldsymbol{w}_{k,i}\}$ by sharing data thoughtfully with the subset of the neighbors that share the same desired model.

5.2.3 Local Labeling

Each agent needs to learn the desired models of its neighbors to proceed with the decision-making process and let the network converge to only one model. In this step instead of only estimating whether two agents have the same desired model or not, the construction involves a local labeling procedure that enables every agent to estimate in real-time how many different models are desired by its neighborhood.

For this purpose, we associate with *each* agent k an $n_k \times n_k$ matrix $\boldsymbol{Y}_i^k$ with entries $\{\boldsymbol{y}_{\ell m}^k(i)\}$ given by:

$$\boldsymbol{y}_{\ell m}^k(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{w}_{m,i-1} - \boldsymbol{w}_{\ell,i-1}\|^2 \leq \beta, \\ 0, & \text{otherwise} \end{cases} \tag{5.48}$$

for some small threshold $\beta > 0$. Whenever $\boldsymbol{y}_{\ell m}^k(i) = 1$, agent k believes at time instant i that its neighbors ℓ and m wish to estimate the same desired model. On account of that the variables $\boldsymbol{w}_{m,i-1}$ and $\boldsymbol{w}_{\ell,i-1}$ which are used in the test (5.48) are presenting the current desired model of agents m and ℓ, respectively. It is clear from (5.48) that the matrix $\boldsymbol{Y}_i^k$ is symmetric and has ones on the diagonal. An example is depicted in Fig. 5.16 where agents having the same inner color observe the same model, while the outer color indicates the current model in which the agent is interested (or towards

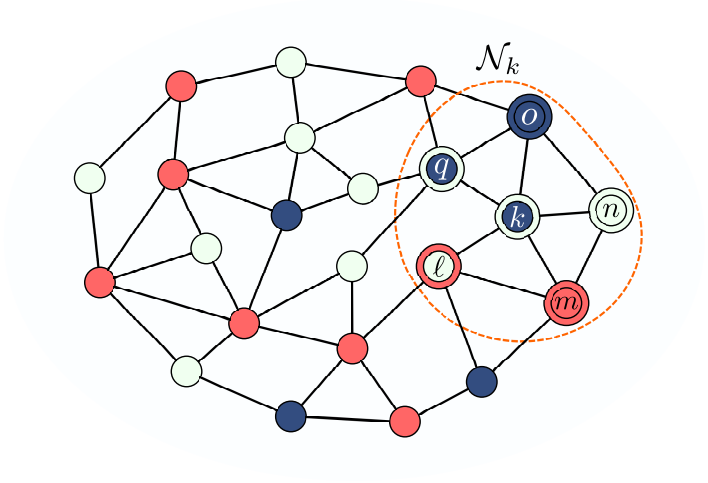

Figure 5.16. Example of an agent k and its neighborhood $\mathcal{N}_k$. The inner color indicates the observing model while the outer one indicates the current desired model.

which the agent is moving in mobile networks). The corresponding matrix $\boldsymbol{Y}_i^k$ has the following entries:

$$\boldsymbol{Y}_i^k = \begin{array}{c} \\ k \\ \ell \\ m \\ n \\ o \\ q \end{array} \begin{array}{c} \begin{array}{cccccc} k & \ell & m & n & o & q \end{array} \\ \left[\begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{array}\right] \end{array}. \tag{5.49}$$

From (5.49) agents that share the same desired model, they have identical columns in matrix $\boldsymbol{Y}_i^k$, namely, if agents m and ℓ have the same desired model at time instant i, this implies that: $[\boldsymbol{Y}_i^k]_{:,m} = [\boldsymbol{Y}_i^k]_{:,\ell}$. We denote the local label of each agent $\ell \in \mathcal{N}_k$ by agent k as $\boldsymbol{l}_\ell^k(i)$. The local label $\boldsymbol{l}_\ell^k(i)$ is updated at each time instant i using the following relation:

$$\boldsymbol{l}_\ell^k(i) = \mathcal{B}([\boldsymbol{Y}_i^k]_{:,\ell}) \tag{5.50}$$

where $\mathcal{B}(\cdot)$ is a function that converts the input sequence from binary to decimal. For the example in (5.49), we have

$$\begin{aligned} \boldsymbol{l}_k^k(i) &= \mathcal{B}(100101) = 37, \\ \boldsymbol{l}_\ell^k(i) &= \mathcal{B}(011000) = 8, \\ \boldsymbol{l}_m^k(i) &= \mathcal{B}(011000) = 8, \\ \boldsymbol{l}_n^k(i) &= \mathcal{B}(100101) = 37, \\ \boldsymbol{l}_o^k(i) &= \mathcal{B}(000010) = 2, \\ \boldsymbol{l}_q^k(i) &= \mathcal{B}(100101) = 37. \end{aligned}$$

We define the number of the different desired models within $\mathcal{N}_k$ at time instant i by $\boldsymbol{C}_k(i)$. After updating matrix $\boldsymbol{Y}_i^k$ and generating the local labels $\{\boldsymbol{l}_\ell^k(i)\}$, agent k counts how many models are desired by its neighborhood to update $\boldsymbol{C}_k(i)$. In the example (5.49), agent k distinguishes at time instant i three desired models, i.e., $\boldsymbol{C}_k(i) = 3$. Where agent k labels these three different models locally by: $\{37, 8, 2\}$.

In addition, agent k determines which model of these $\boldsymbol{C}_k(i)$ models has the maximum number of followers. A *follower* of a model is an agent that wishes to estimate and track this model. We define the largest set of agents belonging to $\mathcal{N}_k$ and following the same desired model at time instant i by $\boldsymbol{\mathcal{Q}}_{k,i}$. In the example, agent k assigns the majority set at time instant i as follows: $\boldsymbol{\mathcal{Q}}_{k,i} = \{k, n, q\}$ which has the label 37 and is repeated three times among the other labels.

5.2.4 Decision-Making Scheme

Using the information provided by matrix $\boldsymbol{Y}_i^k$, agent k can capture how many agents within its neighbors follow the same desired model at time instant i. According to the matrix $\boldsymbol{Y}_i^k$ structure, once agent k and its neighbors have agreed to one common desired model, the corresponding matrix $\boldsymbol{Y}_i^k$ has the following entries:

$$\boldsymbol{Y}_i^k = \begin{array}{c} \\ k \\ \ell \\ m \\ n \\ o \\ q \end{array} \begin{array}{c} \begin{array}{cccccc} k & \ell & m & n & o & q \end{array} \\ \left[\begin{array}{cccccc} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \end{array}. \tag{5.51}$$

We define the degree of the agreement by each agent k among its neighbors $\mathcal{N}_k$ as

$$\boldsymbol{p}_k(i) = \frac{[\boldsymbol{Y}_i^k]_{k,:}\mathbb{1}}{n_k}. \tag{5.52}$$

Equally, having $\boldsymbol{p}_k(i) = 1$ means that agent k and its neighbors have agreed to one common desired model. On the other hand, if $\boldsymbol{p}_k(i) \neq 1$, then the following switching step is applied:

$$\boldsymbol{w}_{k,i-1} = \begin{cases} \boldsymbol{w}_{\ell,i-1}, & \text{if } k \notin \boldsymbol{\mathcal{Q}}_{k,i} \;\forall\; \ell \in \boldsymbol{\mathcal{Q}}_{k,i}, \\ \boldsymbol{w}_{n,i-1}, & \text{if } k \in \boldsymbol{\mathcal{Q}}_{k,i} \text{ AND } \boldsymbol{C}_k(i) = 2 \;\forall n \in \mathcal{N}_k, \\ \boldsymbol{w}_{k,i-1}, & \text{otherwise.} \end{cases} \tag{5.53}$$

The main idea of the switching step is for each agent k to make a new decision or to keep the previous one. The first case of (5.53) implies that agent k does not belong to

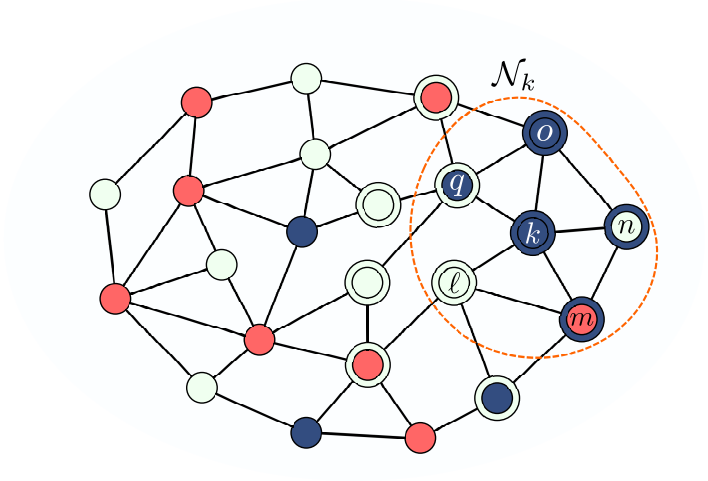

Figure 5.17. Example of the equilibrium case. All agents within $\mathcal{N}_k$ belong to the majority sets among their neighbors.

the majority desired model set $\boldsymbol{\mathcal{Q}}_{k,i}$ at time instant i, therefore, agent k in this case changes its decision and switches into the desired model of the majority set $\boldsymbol{\mathcal{Q}}_{k,i}$. The second case in (5.53) is applied to prevent the balanced situation, i.e., the equilibrium. The problem of an equilibrium might arise when two desired models remain among $\mathcal{N}_k$. In this case, if all agents in $\mathcal{N}_k$ belong to a majority set of one of these different two desired models, this leads to a situation in which no agent in $\mathcal{N}_k$ will change its decision anymore. An example is shown in Fig. 5.17 where the outer color of the agents indicate the desired model, we emphasize only the desired model of agent's k neighbors and their neighbors. Fig. 5.17 shows that all agents within $\mathcal{N}_k$ belong to a majority set and no agent in $\mathcal{N}_k$ will change its decision anymore, e.g. agents q and ℓ belong to the majority set among their neighbors as well as agents k, m, n, and o. Namely,

$$\begin{aligned} &k \in \boldsymbol{\mathcal{Q}}_{k,i}, m \in \boldsymbol{\mathcal{Q}}_{m,i}, n \in \boldsymbol{\mathcal{Q}}_{n,i}, \text{and } o \in \boldsymbol{\mathcal{Q}}_{o,i} \quad (\text{with } z_1^\circ), \\ &\ell \in \boldsymbol{\mathcal{Q}}_{\ell,i} \text{ and } q \in \boldsymbol{\mathcal{Q}}_{q,i} \quad (\text{with } z_2^\circ). \end{aligned}$$

To break the equilibrium, an agent that recognizes these two models picks randomly one of these two desired models.

From (5.53), logically, we can conjecture that the network will probably converge to the most observable model, since the initial desired model by each agent is its own observed model. This fact remains true even with the random switching in the second case of (5.53), because in that case the more repeated desired model within $\mathcal{N}_k$ has the highest probability to be picked.

To proceed with the cooperation and sharing information among the agents within the subset that has the same desired model at time instant i, we define an $N \times N$ matrix

$\boldsymbol{H}_i$. The coefficients $\{\boldsymbol{h}_{\ell k}(i)\}$ are updated after the switching step (5.53) using a test that is quite similar to (5.48) and is applied between each agent k and its neighbors as following:

$$\boldsymbol{h}_{\ell k}(i) = \begin{cases} 1, & \text{if } \|\boldsymbol{w}_{k,i-1} - \boldsymbol{w}_{\ell,i-1}\|^2 \leq \beta, \\ 0, & \text{otherwise.} \end{cases} \tag{5.54}$$

According to matrix $\boldsymbol{H}_i$, each agent knows which subset of its neighbors has the same desired model as it does after the switching step at time instant i. Having $\boldsymbol{h}_{\ell k}(i) = 1$ means that ℓ and k have the same desired model at time instant i. We define an $N \times N$ combination matrix $\boldsymbol{G}_i$ as follows:

$$\boldsymbol{G}_i = \mathcal{F}(\boldsymbol{H}_i) \tag{5.55}$$

where $\mathcal{F}(\cdot)$ is some function which satisfies

$$\boldsymbol{g}_{\ell k}(i) = 0 \quad \text{if} \quad \boldsymbol{h}_{\ell k}(i) = 0, \qquad \sum_{\ell=1}^{N} \boldsymbol{g}_{\ell k}(i) = 1 \tag{5.56}$$

Matrix $\boldsymbol{G}_i$ by itself does not have enough information for proceeding and updating the estimate $\boldsymbol{w}_{k,i}$. The agents still need knowledge about which data to be combined from each neighbor. Therefore, matrix $\boldsymbol{G}_i$ is splitted into two matrices $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$. The weight of the entry $\boldsymbol{g}_{\ell k}(i)$ goes to $\dot{\boldsymbol{a}}_{\ell k}(i)$ if the desired model of agent k at time instant i is the same as ℓ's observed model. Otherwise, $\ddot{\boldsymbol{a}}_{\ell k}(i)$ obtains the weight $\boldsymbol{g}_{\ell k}(i)$. The coefficients $\{\dot{\boldsymbol{a}}_{\ell k}(i)\}$ and $\{\ddot{\boldsymbol{a}}_{\ell k}(i)\}$ for $\ell \in \mathcal{N}_k$ are updated using the following steps:

$$\dot{\boldsymbol{a}}_{\ell k}(i) = \begin{cases} \boldsymbol{g}_{\ell k}(i), & \text{if } \|\boldsymbol{w}_{k,i-1} - \boldsymbol{\psi}_{\ell,i}\|^2 \leq \beta, \\ 0, & \text{otherwise.} \end{cases} \tag{5.57}$$

$$\ddot{\boldsymbol{a}}_{\ell k}(i) = \begin{cases} \boldsymbol{g}_{\ell k}(i), & \text{if } \dot{\boldsymbol{a}}_{\ell k}(i) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.58}$$

In (5.57), the case that $\boldsymbol{\psi}_{\ell,i}$ is close to $\boldsymbol{w}_{\ell,i-1}$ implies that the observed model of agent ℓ is the same as the desired model of agent k at time instant i. The estimate $\boldsymbol{w}_{k,i}$ is updated using (5.47). Algorithm 4 summarizes the steps of the decision-making scheme.

Algorithm 4 (Decision-making scheme)

Initialize $\boldsymbol{A}_0 = \dot{\boldsymbol{A}}_0 = \ddot{\boldsymbol{A}}_0 = \boldsymbol{E}_0 = \boldsymbol{H}_0 = \boldsymbol{G}_0 = I$
Initialize $\boldsymbol{\psi}_0 = \boldsymbol{\phi}_0 = 0$ and $\boldsymbol{p}_0 = 0$
for $i > 0$ **do**

for $k = 1, \ldots, N$ **do**

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{\psi}_{k,i-1}) \tag{5.59}$$

assign $\boldsymbol{w}_{k,0} = \boldsymbol{\psi}_{k,1}$ at $i = 1$
update $\{\boldsymbol{a}_{\ell k}(i)\}$ according to [18]

$$\boldsymbol{\phi}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) \boldsymbol{\psi}_{\ell,i} \tag{5.60}$$

update $\boldsymbol{Y}_i^k$ according to (5.48)
find $\boldsymbol{\mathcal{Q}}_{k,i}$ and $\boldsymbol{C}_k(i)$
update $\boldsymbol{p}_k(i)$ according to (5.52)
if $\boldsymbol{p}_k(i) \neq 1$ **then**
switch $\boldsymbol{w}_{k,i-1}$ according to (5.53)
resend $\boldsymbol{w}_{k,i-1}$
end if
for $\ell \in \mathcal{N}_k$ **do**
update $\{\boldsymbol{h}_{\ell k}(i)\}$ according to (5.54)
update $\{\boldsymbol{g}_{\ell k}(i)\}$ according to (5.55)
update $\{\dot{\boldsymbol{a}}_{\ell k}(i)\}$ according to (5.57)
update $\{\ddot{\boldsymbol{a}}_{\ell k}(i)\}$ according to (5.58)
end for

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \dot{\boldsymbol{a}}_{\ell k}(i) \boldsymbol{\phi}_{\ell,i} + \sum_{\ell=1}^{N} \ddot{\boldsymbol{a}}_{\ell k}(i) \boldsymbol{w}_{\ell,i-1} \tag{5.61}$$

end for
end for

5.2.5 Convergence of Decision-Making Process

Agents achieve the agreement on one desired model over the network when,

$$\boldsymbol{p}_1(i) = \boldsymbol{p}_2(i) = \cdots = \boldsymbol{p}_N(i) = 1. \tag{5.62}$$

The decision-making process can be modeled as a Markov chain with $N+1$ states $\{\mathcal{V}_i\}$ corresponding to the number of agents whose $\boldsymbol{p}_k(i) = 1$. For a given vector $\boldsymbol{p}_{i-1}$ we

have,

$$\mathcal{V}_{i-1} = \sum_{k=1}^{N} \lceil \boldsymbol{p}_k(i-1) \rceil \tag{5.63}$$

where $\lceil \cdot \rceil$ denotes the ceiling operation. Eq. (5.63) aims at counting only $\{\boldsymbol{p}_k(i-1)\}$ that equal to one.

To compute the transition probability $v_{n,m}$ from the state $\mathcal{V}_{i-1} = n$ to the state $\mathcal{V}_i = m$, let the set of vectors

$$\mathcal{P} = \{\lceil p_1 \rceil, \lceil p_2 \rceil, \ldots, \lceil p_N \rceil\}, \tag{5.64}$$

whose entries are either 1 or 0 and add up to n be denoted by $\mathcal{G}_n$. We can write

$$v_{n,m} = \sum_{\boldsymbol{p}_{i-1} \in \mathcal{G}_n} \Pr(\boldsymbol{p}_{i-1}) \sum_{\boldsymbol{p}_i \in \mathcal{G}_m} \prod_{k}^{N} \Pr(\boldsymbol{p}_k(i) | \boldsymbol{p}_k(i-1)) \tag{5.65}$$

where $\Pr(\boldsymbol{p}_{i-1})$ is a priori probability and the probability $\Pr(\boldsymbol{p}_k(i)|\boldsymbol{p}_k(i-1))$ is controlled by the switching step (5.53). The case of $n = N$ implies that for each agent k it holds that

$$\boldsymbol{\mathcal{Q}}_{k,i} \equiv \mathcal{N}_k. \tag{5.66}$$

This leads to the status that the desired model of the whole network will not be changed anymore. The state $\mathcal{V}_i = N$ is called an absorbing state. It has the following probability transitions:

$$v_{N,N} = 1, \;\; v_{N,n} = 0 \;\; \forall n \in \{1, \ldots, N-1\}. \tag{5.67}$$

The transition probability matrix V of the Markov chain can be written as

$$V = \begin{bmatrix} \breve{V} & c \\ 0 & 1 \end{bmatrix} \tag{5.68}$$

where the matrix $\breve{V}$ is of size $N \times N$. According to the Theorem 11.3 in [84, p. 417], it holds that for a Markov process with an absorbing state the probability that the process will be absorbed is 1, namely,

$$\breve{V}^j \to 0 \quad \text{as} \quad j \to \infty. \tag{5.69}$$

Consequently, the Markov chain of the decision-making process converges to the absorbing state $\mathcal{V}_i = N$ regardless of the initial transient state with the convergence rate $\rho(\breve{V})$.

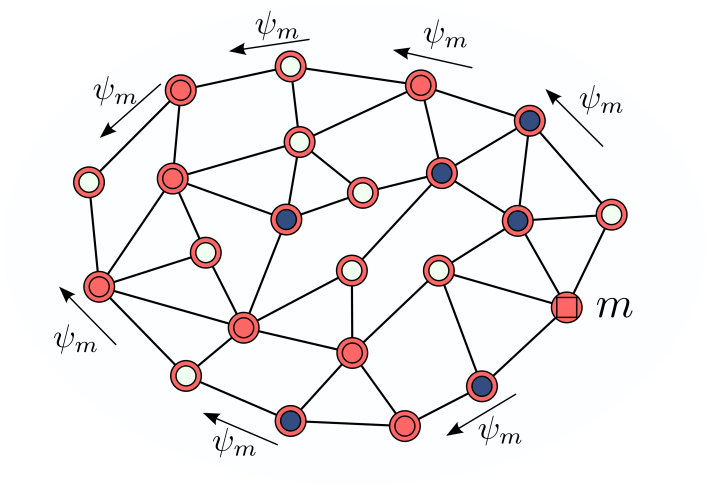

Figure 5.18. Final decision of a network after following the model of the specific agent m. The inner color indicates the observing model while the outer one indicates the desired model. The arrows represent the spreading process of $\boldsymbol{\psi}_{m,i}$ through the network.

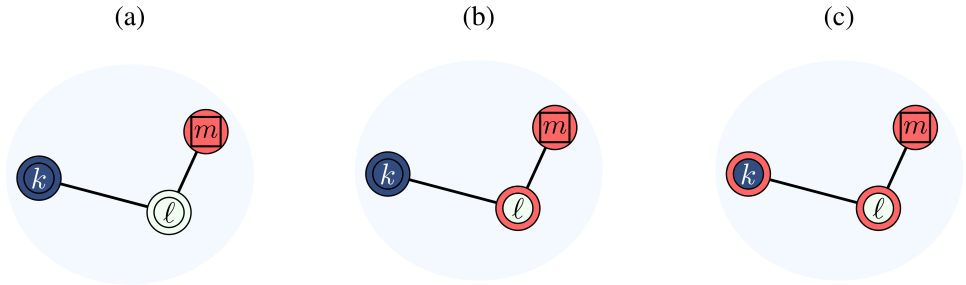

Figure 5.19. Example of the spreading process of $\boldsymbol{\psi}_{m,i}$ from agent m to agent k over time. The inner color indicates the observing model while the outer one indicates the desired model.

5.2.6 Following the Observed Model of a Specific Agent

In this section the goal is to let the whole network follow the observed model of some specific agent m, as shown in Fig. 5.18 where agent m observes the model z_3° (red), therefore, the network converges in a distributed manner to estimate the model z_3°. The first step is to spread the $\boldsymbol{\psi}_{m,i}$ among agents and keep updating it over time. This step aims at having a copy (reference) of $\boldsymbol{\psi}_{m,i}$ by all agents in the network. Agents keep updating the copy of $\boldsymbol{\psi}_{m,i}$ for two reasons. First, to have a more accurate version of the vector $\boldsymbol{\psi}_{m,i}$ which indicates the desired model of the network. Second, to endow the algorithm to work in non-stationary situations, if a drift is happening in agent m's model.

We denote the copy vector of $\boldsymbol{\psi}_{m,i}$ by agent k by $\breve{\boldsymbol{\psi}}_{k,i}$ and denominate it the *anchor*

vector. Agents are informed beforehand about the index m of the specific agent that they should follow its model. If $m \in \mathcal{N}_k$, this implies that agent k receives the anchor vector directly from agent m. If not, i.e., $m \notin \mathcal{N}_k$, then agent k depends on another agent $\ell \in \mathcal{N}_k$ that has already a copy of $\boldsymbol{\psi}_{m,i}$ to provide it with $\boldsymbol{\psi}_{m,i}$. Agent k stores the index of this source agent. The index of the source agent of agent k is denoted by $\boldsymbol{s}_k(i)$. Note that the anchor vector $\breve{\boldsymbol{\psi}}_{k,i}$ is not the final estimate of the desired model.

The circulation process of $\boldsymbol{\psi}_{m,i}$ in a distributed manner needs the cooperation among agents. In case that agent k has no direct link to reach data from agent m, i.e., $m \notin \mathcal{N}_k$, agent k gets one of the $\breve{\boldsymbol{\psi}}_{\ell,i-1}$ provided that $\boldsymbol{s}_\ell(i) \neq 0$. If $\boldsymbol{s}_\ell(i) \neq 0$ this implies that agent ℓ has already a source to update its $\breve{\boldsymbol{\psi}}_{\ell,i}$, regardless whether $m \in \mathcal{N}_\ell$ or not. In other words, $\boldsymbol{s}_\ell(i) \neq 0$ means that agent ℓ finds a direct or indirect link to agent m. Therefore, it is important for each agent k to store the agent's index of its source that agent k depends on to update its anchor vector. An example is shown in Fig. 5.19 where $m \in \mathcal{N}_\ell$ but $m \notin \mathcal{N}_k$. First, the anchor vectors and the source agents for agents k and ℓ at time instant $i = 0$ (Fig. 5.19(a)) are given, respectively, by

$$\breve{\boldsymbol{\psi}}_{k,0} = 0, \boldsymbol{s}_k(0) = 0, \breve{\boldsymbol{\psi}}_{\ell,0} = 0, \boldsymbol{s}_\ell(0) = 0.$$

The anchor vectors and the source agents for agents k and ℓ at time instants $i = \{1, 2\}$ (Fig. 5.19(b) and (c)) are given, respectively, by

$$\begin{aligned}
&\breve{\boldsymbol{\psi}}_{k,1} = 0, \boldsymbol{s}_k(1) = 0, \breve{\boldsymbol{\psi}}_{\ell,1} = \boldsymbol{\psi}_{m,1}, \boldsymbol{s}_\ell(1) = m, \\
&\breve{\boldsymbol{\psi}}_{k,2} = \breve{\boldsymbol{\psi}}_{\ell,1}, \boldsymbol{s}_k(2) = \ell, \breve{\boldsymbol{\psi}}_{\ell,2} = \boldsymbol{\psi}_{m,2}, \boldsymbol{s}_\ell(2) = m.
\end{aligned}$$

Agents update their anchor vectors $\{\breve{\boldsymbol{\psi}}_{k,i}\}$ at each time instant i by the following step:

$$\breve{\boldsymbol{\psi}}_{k,i} = \begin{cases} \boldsymbol{\psi}_{m,i}, & \text{if } m \in \mathcal{N}_k, \\ \breve{\boldsymbol{\psi}}_{\ell,i-1}, & \text{if } \ell \in \mathcal{N}_k \text{ AND } \boldsymbol{s}_k(i) = 0 \text{ AND } \boldsymbol{s}_\ell(i) \neq 0, \\ \breve{\boldsymbol{\psi}}_{\ell,i-1}, & \text{if } \ell \in \mathcal{N}_k \text{ AND } \boldsymbol{s}_k(i) = \ell, \\ \breve{\boldsymbol{\psi}}_{k,i-1}, & \text{otherwise} \end{cases} \tag{5.70}$$

where $\breve{\boldsymbol{\psi}}_{m,i} = \boldsymbol{\psi}_{m,i}$ for agent m itself. The source of the anchor vector is updated simultaneously as follows:

$$\boldsymbol{s}_k(i) = \begin{cases} m, & \text{if } m \in \mathcal{N}_k, \\ \ell, & \text{if } \boldsymbol{s}_k(i) = 0 \text{ AND } \boldsymbol{s}_\ell(i) \neq 0, \\ \boldsymbol{s}_k(i-1), & \text{otherwise.} \end{cases} \tag{5.71}$$

Similarly to the previous section, the next step is to update the coefficients $\{\boldsymbol{h}_{\ell k}(i)\}$ using the following test:

$$\boldsymbol{h}_{\ell k}(i) = \begin{cases} 1, & \text{if } \boldsymbol{s}_\ell(i) \neq 0 \text{ AND } \boldsymbol{s}_k(i) \neq 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.72}$$

Again, having $\boldsymbol{s}_k(i) \neq 0$ leads to the situation that agent k has the anchor vector and has been informed about the decision of the network, therefore, agent k can start sharing information with the other agents whose $\boldsymbol{s}_\ell(i) \neq 0$ as well to estimate the desired model. The matrix $\boldsymbol{G}_i$ will be generated using (5.55). Agents update the coefficients of both matrices $\dot{\boldsymbol{A}}_i$ and $\ddot{\boldsymbol{A}}_i$ using the following steps:

$$\dot{\boldsymbol{a}}_{\ell k}(i) = \begin{cases} \boldsymbol{g}_{\ell k}(i), & \text{if } \|\breve{\boldsymbol{\psi}}_{k,i} - \boldsymbol{\psi}_{\ell,i}\|^2 \leq \beta, \\ 0, & \text{otherwise.} \end{cases} \tag{5.73}$$

$$\ddot{\boldsymbol{a}}_{\ell k}(i) = \begin{cases} \boldsymbol{g}_{\ell k}(i), & \text{if } \dot{\boldsymbol{a}}_{\ell k}(i) = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.74}$$

Then, the estimate $\boldsymbol{w}_{k,i}$ is updated using Eq. (5.47). According to (5.73) and (5.47) agent k combines $\boldsymbol{\phi}_{\ell,i}$ if the desired model of the network (which is represented by the anchor vector $\breve{\boldsymbol{\psi}}_{k,i}$ of agent k) is close to the observed model of agent ℓ that is represented by $\boldsymbol{\psi}_{\ell,i}$. Algorithm 5 summarizes the steps of the algorithm for following the observed model of a specific agent m.

5.2.7 Simulation Results

5.2.7.1 Static Network

We consider a connected network with 80 randomly distributed agents. The agents observe data originating from $C = 3$ different models, each model $z_j^\circ \in \mathbb{R}^{M\times 1}$ is generated as follows: $z_j^\circ = [z_{r_1}, \ldots, z_{r_M}]^\intercal$ where $z_{r_m} \in [1, -1]$, $M = 2$. The assignment of the agents to models is random. The maximum number of neighbors is $n_k = 7$. We set $\{\alpha, \beta, \nu, \mu\} = \{0.04, 0.08, 0.005, 0.01\}$. We use the uniform combination policy to generate the coefficients $\{\boldsymbol{a}_{\ell k}(i)\}$ and $\{\boldsymbol{g}_{\ell k}(i)\}$.

Figure 5.20 shows the statistical profile of the regressors and noise across the agents. The regressors are of size $M = 2$ zero-mean Gaussian, independent in time and space, and have diagonal covariance matrices $R_{u,k}$. Figure 5.21 shows the topology of one of 100 Monte Carlo experiments. Agents having the same inner color observe the same model, while the outer color indicates the desired model at steady-state.

The transient network mean-square deviation (MSD) regarding each observed model z_j° at each time instant i is defined by

$$\text{MSD}_j(i) \triangleq \frac{1}{|\mathcal{C}_j|} \sum_{k\in\mathcal{C}_j} ||z_j^\circ - \boldsymbol{\phi}_{k,i}||^2 \tag{5.78}$$

Algorithm 5 (Following the observed model of a specific agent)

Initialize $\boldsymbol{A}_0 = \dot{\boldsymbol{A}}_0 = \ddot{\boldsymbol{A}}_0 = \boldsymbol{E}_0 = \boldsymbol{H}_0 = \boldsymbol{G}_0 = I$
Initialize $\boldsymbol{\psi}_0 = \breve{\boldsymbol{\psi}}_0 = \boldsymbol{\phi}_0 = 0$ and $\boldsymbol{s}_0 = 0$

for $i > 0$ **do**

for $k = 1, \ldots, N$ **do**

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{\psi}_{k,i-1} + \mu_k \boldsymbol{u}_{k,i}^{\mathsf{T}} (\boldsymbol{d}_k(i) - \boldsymbol{u}_{k,i} \boldsymbol{\psi}_{k,i-1}) \tag{5.75}$$

assign $\boldsymbol{w}_{k,0} = \boldsymbol{\psi}_{k,1}$ at $i = 1$
update $\{\boldsymbol{a}_{\ell k}(i)\}$ according to [18]

$$\boldsymbol{\phi}_{k,i} = \sum_{\ell=1}^{N} \boldsymbol{a}_{\ell k}(i) \boldsymbol{\psi}_{\ell,i} \tag{5.76}$$

update $\breve{\boldsymbol{\psi}}_{k,i}$ according to (5.70)
update $\boldsymbol{s}_k(i)$ according to (5.71)

for $\ell \in \mathcal{N}_k$ **do**

update $\{\boldsymbol{h}_{\ell k}(i)\}$ according to (5.72)
update $\{\boldsymbol{g}_{\ell k}(i)\}$ according to (5.55)
update $\{\dot{\boldsymbol{a}}_{\ell k}(i)\}$ according to (5.73)
update $\{\ddot{\boldsymbol{a}}_{\ell k}(i)\}$ according to (5.74)

end for

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} \dot{\boldsymbol{a}}_{\ell k}(i) \boldsymbol{\phi}_{\ell,i} + \sum_{\ell=1}^{N} \ddot{\boldsymbol{a}}_{\ell k}(i) \boldsymbol{w}_{\ell,i-1} \tag{5.77}$$

end for

end for

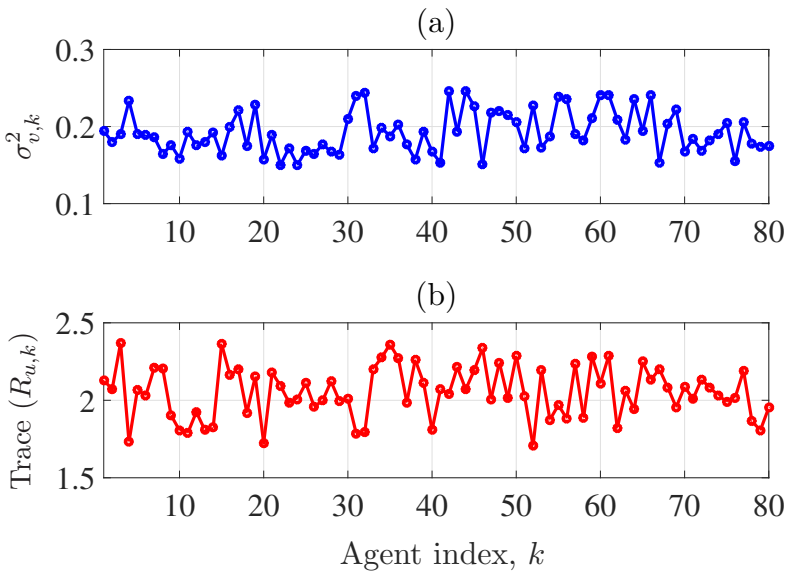

Figure 5.20. Statistical noise and signal profiles over the network.

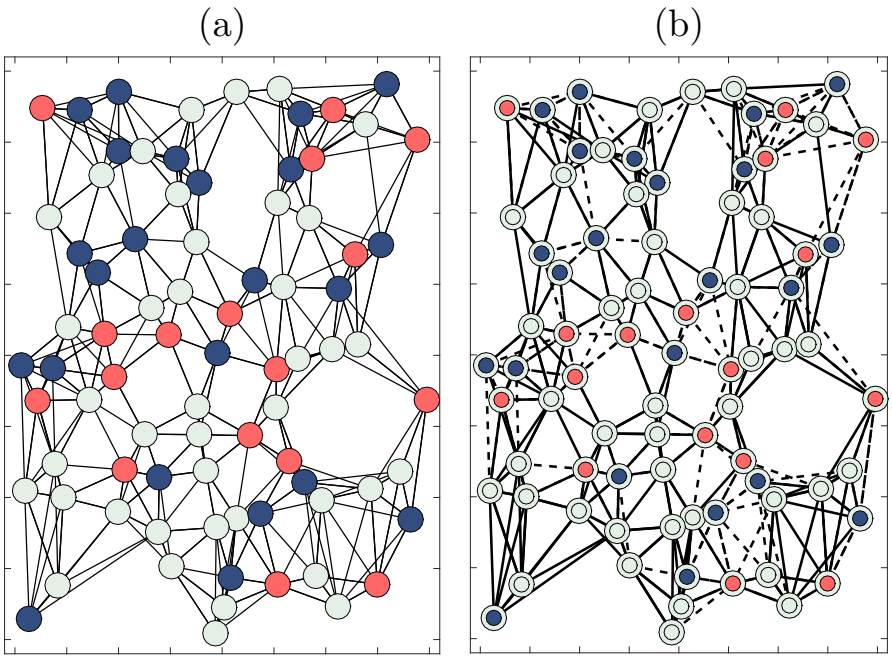

Figure 5.21. Network topology (a) and final decision of the agents where the bold (dashed) links represent $\{\dot{\boldsymbol{a}}(i)\}$ ($\{\ddot{\boldsymbol{a}}(i)\}$) at steady-state (b).

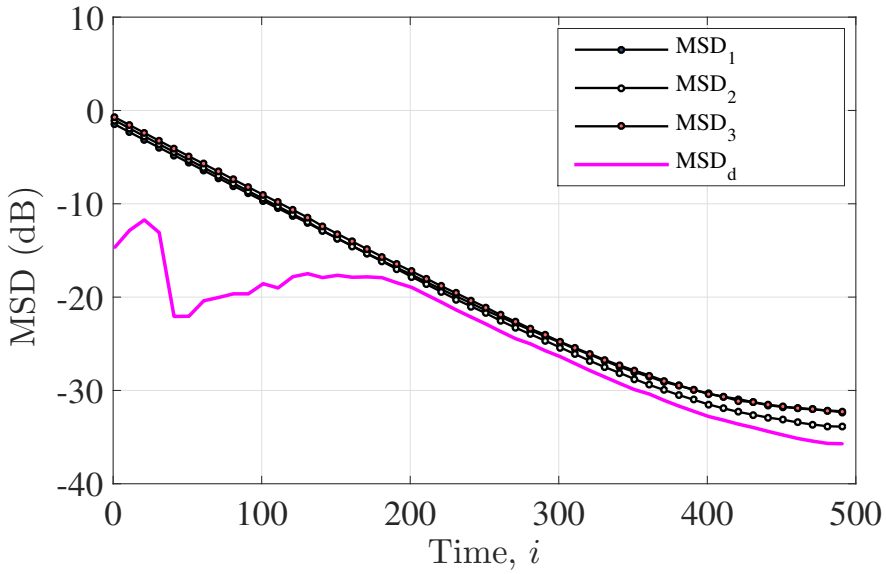

Figure 5.22. Transient mean-square deviation (MSD).

where $j = 1, \ldots, C$ and each MSD_j is computed for agents belonging to $\mathcal{C}_j$. The transient network mean-square deviation (MSD) for the whole network regarding the desired model at each time instant i is defined by

$$\text{MSD}_{\text{d}}(i) \triangleq \frac{1}{N} \sum_{k=1}^{N} ||z_d^{\circ} - \boldsymbol{w}_{k,i}||^2 \tag{5.79}$$

where z_d° is the desired model when the whole network agrees on one common desired model, i.e., $\text{MSD}_{\text{d}}(i)$ is only computed at the instants when $\{\boldsymbol{p}_k(i)\} = 1$ for all agents. Figure 5.22 depicts the simulated transient mean-square deviation (MSD) of the network for all observed models and for the network desired model. Table 5.2 displays the success rate of the decision-making to agree on one model for different numbers of observed model, $C \in \{2, 3, 4, 5\}$. The proposed strategy provides almost 100% success rate.

C	2	3	4	5
Success rate	99%	98%	99%	99%

Table 5.2. Decision-making success rate for different C.

Regarding the application of following the observed model of a specific agent m, Fig. 5.23 shows the topology of one case of 100 different experiments. Agents are observing $C = 4$ different models. The agent $m = 10$, which is presented by a square, is the specific

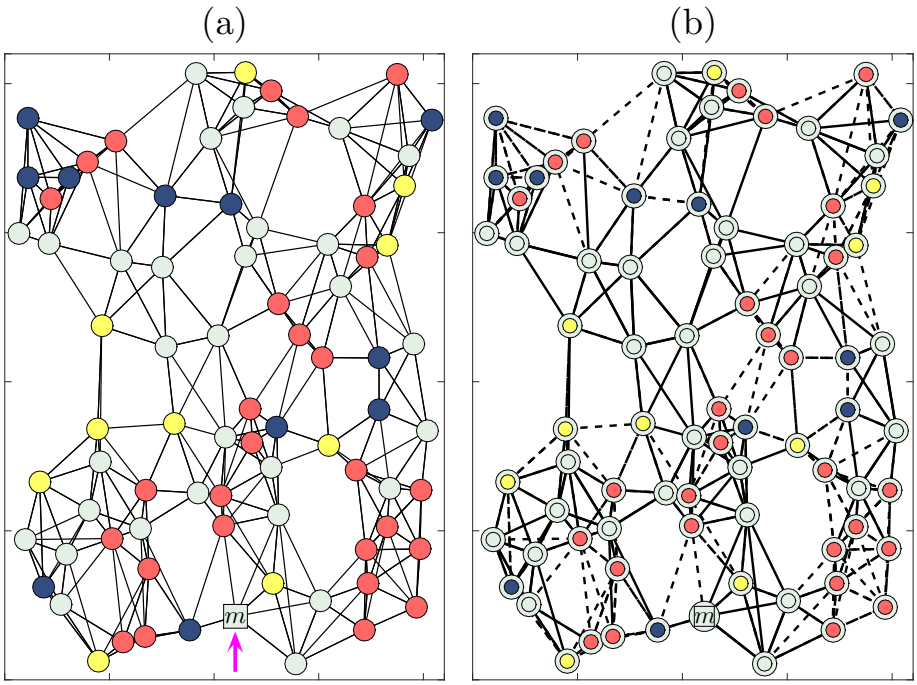

Figure 5.23. Network topology (a) and final decision of the agents to follow the model of agent m where the bold (dashed) links represent $\{\dot{\boldsymbol{a}}(i)\}$ ($\{\ddot{\boldsymbol{a}}(i)\}$) at steady-state (b).

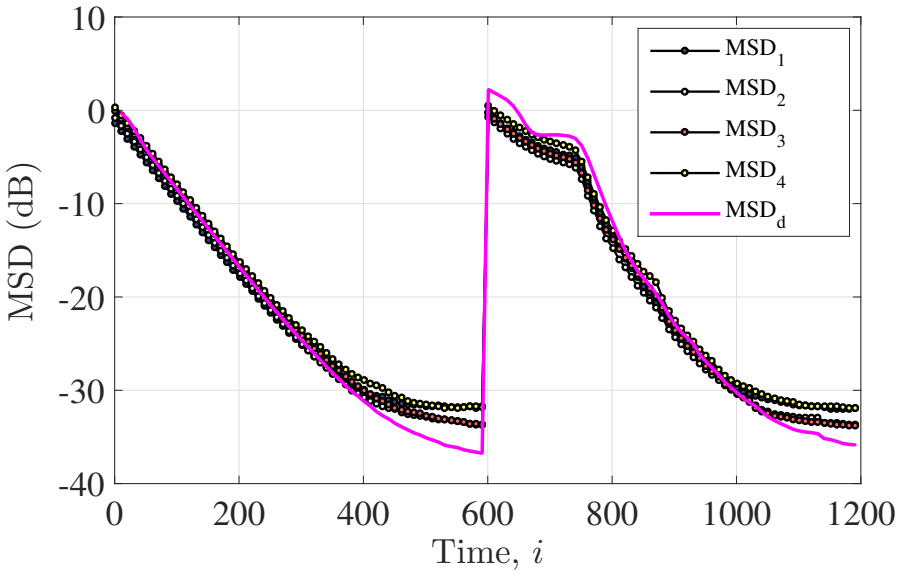

Figure 5.24. Transient mean-square deviation (MSD).

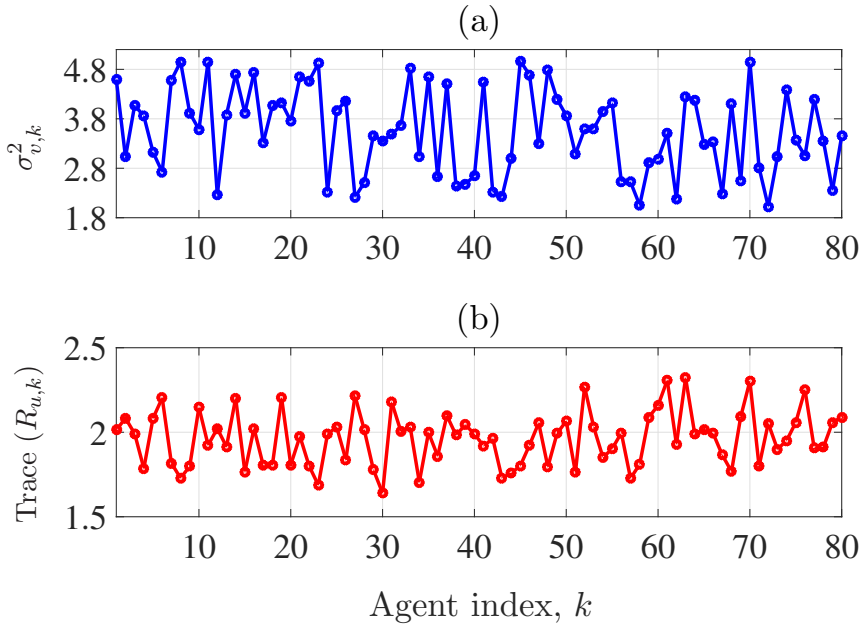

Figure 5.25. Statistical noise and signal profiles over the mobile network.

agent whose observed model the whole network wishes to follow. Figure 5.24 shows the transient mean-square deviation MSD of 100 different experiments when a change in the model assignments occurs suddenly at time instant $i = 600$. The success rate of the decision-making to agree on the observed model of agent m is always 100%.

5.2.7.2 Mobile Network

We consider a network with 80 randomly distributed mobile agents [42]. The agents observe data originating from four different models (sources) $C = 4$, where $w_{r_m} \in [50, -50]$. The objective of the network is to have all agents track and move towards only one model (source). Figure 5.25 shows the statistical profile of the regressors and noise across the agents. Every agent k updates its location according to the motion mechanism proposed in [21].

Figure 5.26 shows the maneuver of the agents over time where the models (sources) are represented by squares. Figure 5.27 represents the transient network mean-square deviation (MSD) obtained by averaging over 100 independent Monte Carlo experiments.

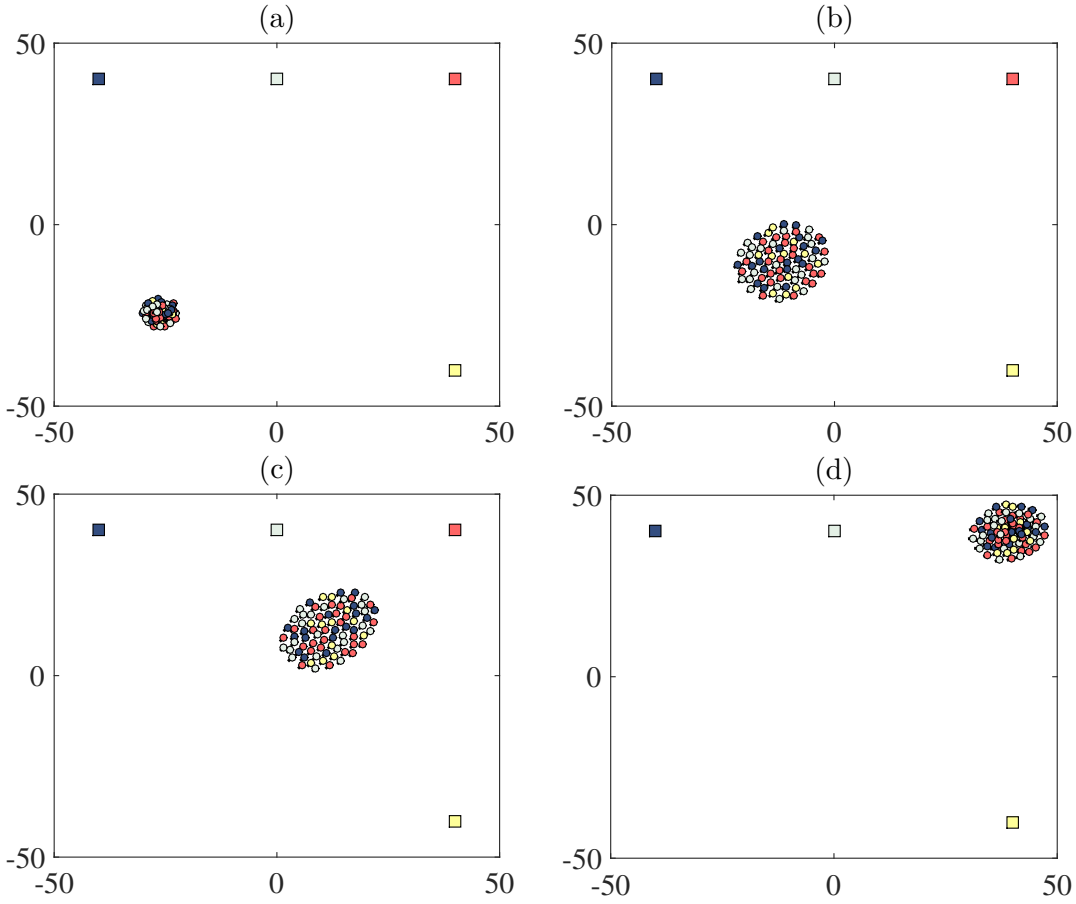

Figure 5.26. Maneuver of the agents with four sources in time instants i=1 (a), i=200 (b), i=500 (c), and i=1000 (d). The unit length of the x- and y-axis is the body length of the agents.

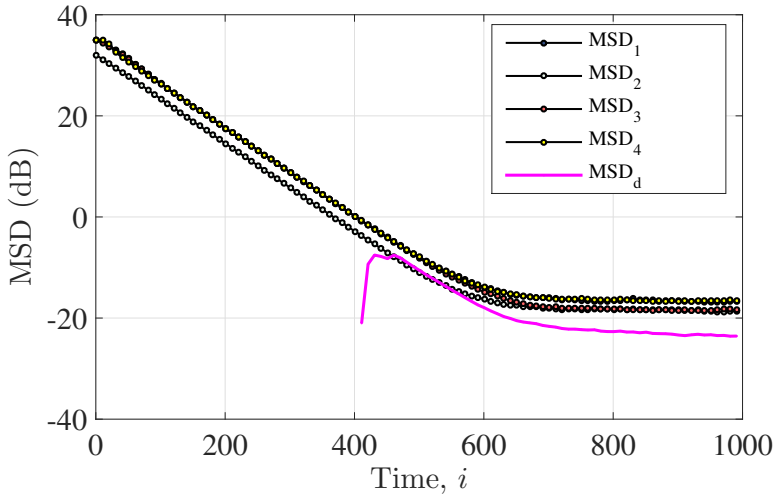

Figure 5.27. Transient mean-square deviation (MSD) of the mobile network.

Chapter 6

Conclusions

'What you seek is seeking you'

Jalaluddin Rumi

In this thesis, we proposed several decentralized approaches that mimic and simulate interesting collective behaviors of animal groups. We focus on multi-task networks where agents are interested in different objectives.

In **Chapter 3** we proposed a distributed algorithm that carries out the tasks of estimation and clustering simultaneously with exponentially decaying error probabilities for false decisions. We showed how agents choose subset of their neighbors to cooperate with and turn off suspicious links. The simulations illustrate the performance of the proposed strategy compared with other related works. To make the problem more flexible and efficient we have not assumed connected clusters. Therefore, we proposed an additional step to enhance the performance by linking, as much as possible, the agents that belong to the same cluster and do not have direct links to connect them. Obviously, for a special case with connected clusters there is no need for the linking procedure.

The proposed decentralized partitioning technique in **Chapter 4** aims to implement a dynamic multi-task network using adaptation and learning in the presence of informed and uninformed agents. The algorithm ensures fair partitioning to distribute the uninformed agents among the groups that are interested in different tasks. Moreover, the decentralized technique has a self-organizing feature that endows the network with learning ability in stationary and non-stationary environments. We applied the proposed technique in both static and mobile networks and showed that the group sizes match well the centralized partitioning size.

A distributed decision-making approach over mobile adaptive networks is studied in **Chapter 5**. We showed that our proposed clustering technique reduces the error that affects the decision-making process. We added a new term to the motion equation to ensure that the nodes move coherently without fragmentation. Simulation results showed that the proposed strategy is insensitive to the initial network location with almost 100% success rate of decision-making.

Furthermore, in **Chapter 5** we proposed a distributed algorithm that allows agents in multi-task networks to follow only one common model. Agents use a local labeling step to distinguish the multiple desired models of their neighbors. The decision-making process depends on the majority of the neighbors that decided to follow the same model. We showed that after sufficient iterations the decision-making process converges to an agreement on one model over the network. In addition, we investigated another technique that guides the whole network to follow a specific agent's model in a distributed way. Simulation results showed that the proposed strategy provides almost 100% success rate of decision-making in both static and mobile networks.

As open problems and for future work, it will be useful to provide the convergence rate of the decision-making process of the approach proposed in **Chapter 5**. Moreover, having a linking mechanism to link agents for more than only one step as it was proposed in **Chapter 3** might provide a better estimation accuracy.

Appendix

A.1 Diffusion Strategies With More Cooperation

Let us define the $[N \times N]$ right-stochastic matrix C where the coefficients $\{c_{\ell k}\}$ are non-negative scalars that satisfy the following conditions for all agents $k = 1, 2, \ldots, N$:

$$c_{\ell k} \geq 0, \ \sum_{k=1}^{N} c_{\ell k} = 1, \ \text{ and } c_{\ell k} = 0 \text{ if } \ell \notin \mathcal{N}_k. \tag{A.1}$$

Each row of C should add up to one, i.e.,

$$C\mathbb{1} = \mathbb{1}. \tag{A.2}$$

The combination coefficients $\{c_{\ell k}\}$ aggregate the gradient terms from neighbors. The ATC algorithm with more cooperation among agents has the following form:

$$\boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu_k \sum_{\ell=1}^{N} c_{\ell k} \widehat{\nabla_{w^\intercal} J_\ell}(\boldsymbol{w}_{k,i-1}) \tag{A.3}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{N} a_{\ell k} \boldsymbol{\psi}_{\ell,i}. \tag{A.4}$$

Likewise, the CTA algorithm is given by

$$\boldsymbol{\psi}_{k,i-1} = \sum_{\ell=1}^{N} a_{\ell k} \boldsymbol{w}_{\ell,i-1} \tag{A.5}$$

$$\boldsymbol{w}_{k,i} = \boldsymbol{\psi}_{k,i-1} - \mu_k \sum_{\ell=1}^{N} c_{\ell k} \widehat{\nabla_{w^\intercal} J_\ell}(\boldsymbol{\psi}_{k,i-1}). \tag{A.6}$$

A.2 Properties of Stochastic Matrices

An $N \times N$ matrix A is called a doubly-stochastic matrix if its column entries and row entries add up to one

$$\mathbb{1}A^\intercal = \mathbb{1}, \ A\mathbb{1} = \mathbb{1}. \tag{A.7}$$

The following properties hold for any doubly-stochastic matrix A:

1. The spectral radius of A is equal to one,

$$\rho(A) = 1. \tag{A.8}$$

2. All eigenvalues of A lie inside the unit disc.

3. Matrix A may have multiple eigenvalues with magnitude equal to one.

4. If A is a *primitive matrix*, then matrix A has a single eigenvalue that is equal to one.

List of Acronyms

MSE	Mean Square Error
EMSE	Excess Mean Square Error
ER	Excess Risk
MSD	Mean Square Deviation
LMS	Least Mean Square
ATC	Adapt-then-Combine
CTA	Combine-then-Adapt
i.i.d.	independent and identically distributed
pdf	probability density function

List of Symbols

$\boldsymbol{x}$	Boldface notation denotes random variables
X	Capital letters denote matrices
x	Small letters denote vectors or scalars
α	Greek letters denote scalars
$x(i)$	Scalar quantities are indexed using parenthesis
x_i	Vector quantities are indexed using subscripts
x_k	$k-$th element of vector x
$x_{k,i}$	$k-$th element of vector x at time instant i
$x_{\ell k}$	Element of matrix X
$\big[X\big]_{k,:}$	Vector corresponding to the $k-$th row of matrix X
$\big[X\big]_{:,k}$	Vector corresponding to the $k-$th column of matrix X
$X > 0$	Positive definite matrix
$X \geq 0$	Non-negative definite matrix
x°	True value of the estimate x_i
$\tilde{x}_i$	Error vector at time instant i
$\bar{x_i}$	Mean value

$\mathbb{1}$	Vector of ones
I	Identity matrix
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^M$	Set of vectors of size M on $\mathbb{R}$
$\mathbb{E}(\cdot)$	Expectation operator
$(\cdot)^\intercal$	Transpose of a vector or matrix
$(\cdot)^*$	Conjugate of a scalar, vector, or matrix
$(\cdot)^{-1}$	Inverse of a matrix
$\mathrm{Tr}(\cdot)$	Trace of a matrix
$\rho(\cdot)$	Spectral radius of a matrix
$\lambda_{\min}(\cdot)$	Minimum eigenvalue of a matrix
$\lambda_{\max}(\cdot)$	Maximum eigenvalue of a matrix
$\|\cdot\|$	Euclidean norm or 2-norm of a vector
$\|\cdot\|_\infty$	Maximum absolute row sum of a matrix
$\|x\|^2_\Sigma$	Weighted squared Euclidean norm of x, $x^\intercal \Sigma x$

$\text{diag}\{\cdot\}$	Returns a diagonal matrix when the argument is a vector, or returns a vector containing the elements of the main diagonal when the argument is a matrix
$\text{col}\{\cdot\}$	Column vector formed by stacking the input entries
$\lvert\cdot\rvert$	Absolute value of a scalar
$\lfloor\cdot\rceil$	Rounds to the nearest integer
$\exp(\cdot)$	Natural exponential function
$\log(\cdot)$	Natural logarithm function
e	Base of the natural logarithm
$\mathcal{O}(\cdot)$	Complexity order of the argument
$\arg\max_x y$	Returns the value of x that maximizes y
$\arg\min_x y$	Returns the value of x that minimizes y
$x \triangleq y$	x is defined as y
$x \equiv y$	x is identically equal to y
$X \otimes Y$	Kronecker product of X and Y
$\limsup_{i\to\infty} x_i$	Limit superior of the sequence x_i as i approaches infinity
$\nabla_x J(x)$	Gradient of scalar or vector function $J(x)$ with respect to vector x
$\mathbb{N}(0,\sigma^2)$	Gaussian distribution with mean 0 and variance σ^2
$Q(\cdot)$	Right-tail Gaussian probability function, $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{t^2}{2}}\,dt$

μ	Step-size
$\mu_{\max}$	Maximum step-size over the network
w°	Optimal weight vector
$d(i)$	Reference signal at time instant i
u_i	Regressor at time instant i (row vector)
$v(i)$	Measurement noise at time instant i
R_u	Covariance matrix of the regression data
σ_v^2	Noise variance
w_i	Weight estimate at time instant i
$\tilde{w}_i$	Weight-error vector at time instant i
$\mathcal{N}_k$	Neighborhood of agent k
$\mathcal{N}_k^-$	Neighborhood of agent k excluding k itself
n_k	Neighborhood size of agent k
A	Combination matrix
$a_{\ell k}$	Weight used by agent k to scale the data it received from agent ℓ

$d_k(i)$	Reference signal of agent k at time instant i
$u_{k,i}$	Regressor of agent k at time instant i (row vector)
$v_k(i)$	Measurement noise of agent k at time instant i
$w_{k,i}$	Weight estimate of agent k at time instant i
$\widetilde{w}_{k,i}$	Weight-error vector of agent k at time instant i
$R_{u,k}$	Covariance matrix of the regression data of agent k
$\sigma^2_{v,k}$	Noise variance of agent k

Bibliography

[1] J. L. Fernndez-Marquez, *Bio-inspired Mechanisms for Self-organising Systems*. CSIC Press, 2012.

[2] B. K. Panigrahi, Y. Shi, and M.-H. Lim, *Handbook of Swarm Intelligence: Concepts, Principles and Applications*. Springer Publishing Company, Incorporated, 1st ed., 2011.

[3] D. W. Corne, A. Reynolds, and E. Bonabeau, *Swarm Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[4] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, pp. 25–34, August 1987.

[5] A. H. Sayed, *Adaptive Filters*. NJ: Wiley, 2008.

[6] A. H. Sayed, "Adaptive networks," in *Proc. IEEE*, vol. 102, pp. 460–497, April 2014.

[7] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing* (R. Chellapa and S. Theodoridis, eds.), vol. 3, pp. 323–454, Academic Press, Elsevier, 2014.

[8] A. Nedic and A. Ozdaglar, *Cooperative distributed multi-agent optimization*, pp. 340–386. Cambridge University Press, 2009.

[9] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Processing*, vol. 58, pp. 1035–1048, March 2010.

[10] A. H. Sayed, S. Y. Tu, J. Chen, X. Zhao, and Z. J. Towfic, "Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior," *IEEE Signal Processing Magazine*, vol. 30, pp. 155–171, May 2013.

[11] J. Chen and A. H.Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Trans. Signal Processing*, vol. 60, pp. 4287–4305, August 2012.

[12] F. S. Cattivelli and A. H. Sayed, "Multi-level diffusion adaptive networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Taipei, Taiwan), pp. 2789–2792, April 2009.

[13] J. Chen, S. K. Ting, C. Richard, and A. H. Sayed, "Group diffusion LMS," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Shanghai, China), pp. 4925–4929, March 2016.

[14] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Processing*, vol. 56, pp. 3122–3136, July 2008.

[15] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, vol. 3, (Honolulu, USA), pp. 917–920, April 2007.

[16] J. Chen and A. H. Sayed, "Performance of diffusion adaptation for collaborative optimization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Kyoto, Japan), pp. 3753–3756, March 2012.

[17] S. Khawatmi, A. H. Sayed, and A. M. Zoubir, "Decentralized clustering and linking by networked agents," *IEEE Trans. Signal Processing*, vol. 65, pp. 3526–3537, July 2017.

[18] S. Khawatmi, A. M. Zoubir, and A. H. Sayed, "Decentralized clustering over adaptive networks," in *Proc. 23rd European Signal Processing Conference (EUSIPCO)*, (Nice, France), pp. 2745–2749, September 2015.

[19] S. Khawatmi, "Signal processing over multi-task networks," in *Proc. 23rd European Signal Processing Conference (EUSIPCO)*, (Nice, France), p. 1562, September 2015.

[20] S. Khawatmi and A. M. Zoubir, "Decentralized partitioning over adaptive networks," in *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, (Vietri sul Mare, Salerno, Italy), pp. 1–6, September 2016.

[21] S. Khawatmi, X. Huang, and A. M. Zoubir, "Distributed decision-making over mobile adaptive networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (New Orleans, USA), pp. 3864–3868, March 2017.

[22] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Found. Trends in Mach. Learn.*, vol. 7, pp. 311–801, July 2014.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[24] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[25] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks* (D. Saad, ed.), Cambridge, UK: Cambridge University Press, 1998.

[26] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, pp. 56–69, July 2006.

[27] S. Chouvardas, K. Slavakis, and S. Theodoridis, "Adaptive robust distributed learning in diffusion sensor networks," *IEEE Trans. Signal Processing*, vol. 59, pp. 4692–4707, October 2011.

[28] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proc. of the 3rd International Symposium on Information Processing in Sensor Networks*, (New York, USA), pp. 20–27, April 2004.

[29] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multiagent optimization," *IEEE Trans. Autom. Control*, vol. 54, pp. 48–61, January 2009.

[30] A. Bertrand and M. Moonen, "Distributed adaptive node-specific signal estimation in fully connected sensor networks, Part I: Sequential node updating," *IEEE Trans. Signal Processing*, vol. 58, pp. 5277–5291, October 2010.

[31] S. Al-Sayed, A. M. Zoubir, and A. H. Sayed, "Robust adaptation in impulsive noise," *IEEE Trans. Signal Processing*, vol. 64, pp. 2851–2865, June 2016.

[32] R. Bitmead, "Convergence in distribution of LMS-type adaptive parameter estimates," *IEEE Trans. Autom. Control*, vol. 28, pp. 54–60, January 1983.

[33] N. Bogdanovic, J. Plata-Chaves, and K. Berberidis, "Distributed diffusion-based LMS for node-specific parameter estimation over adaptive networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Florence, Italy), pp. 7223–7227, May 2014.

[34] J. Fernandez-Bes, J. Arenas-Garcia, and A. H. Sayed, "Adjustment of combination weights over adaptive diffusion networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Florence, Italy), pp. 1–5, May 2014.

[35] S. Y. Tu and A. H. Sayed, "Optimal combination rules for adaptation and learning over networks," in *Proc. 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, (San Juan, Puerto Rico), pp. 317–320, December 2011.

[36] S. Y. Tu and A. H. Sayed, "On the effects of topology and node distribution on learning over complex adaptive networks," in *Proc. Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, USA), pp. 1166–1171, November 2011.

[37] Z. J. Towfic, J. Chen, and A. H. Sayed, "On the generalization ability of distributed online learners," in *Proc. IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, September 2012.

[38] J. Chen and A. H. Sayed, "On the limiting behavior of distributed optimization strategies," in *Proc. 50th Annual Allerton Conference on Communication, Control, and Computing*, (Monticello, USA), pp. 1535–1542, October 2012.

[39] J. Chen and A. H. Sayed, "Distributed Pareto optimization via diffusion adaptation," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, pp. 205–220, April 2013.

[40] S. Y. Tu and A. H. Sayed, "Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks," *IEEE Trans. Signal Processing*, vol. 60, pp. 6217–6234, December 2012.

[41] X. Zhao and A. H. Sayed, "Learning over social networks via diffusion adaptation," in *Proc. Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, USA), pp. 709–713, November 2012.

[42] S. Y. Tu and A. H. Sayed, "Mobile adaptive networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, pp. 649–664, August 2011.

[43] S. Y. Tu and A. H. Sayed, "Tracking behavior of mobile adaptive networks," in *Proc. Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, CA), pp. 698–702, November 2010.

[44] S. Y. Tu and A. H. Sayed, "Cooperative prey herding based on diffusion adaptation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Prague, Czech Republic), pp. 3752–3755, May 2011.

[45] F. S. Cattivelli and A. H. Sayed, "Modeling bird flight formations using diffusion adaptation," *IEEE Trans. Signal Processing*, vol. 59, pp. 2038–2051, May 2011.

[46] F. S. Cattivelli and A. H. Sayed, "Self-organization in bird flight formations using diffusion adaptation," in *Proc. 3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, (Aruba, Dutch Antilles, Netherlands), pp. 49–52, December 2009.

[47] J. Li, S. Y. Tu, and A. H. Sayed, "Honeybee swarming behavior using diffusion adaptation," in *Proc. IEEE Digital Signal Processing Workshop*, (Sedona, USA), pp. 249–254, January 2011.

[48] J. Chen, X. Zhao, and A. H. Sayed, "Bacterial motility via diffusion adaptation," in *Proc. Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, USA), pp. 1930–1934, November 2010.

[49] J. Chen and A. H. Sayed, "Bio-inspired cooperative optimization with application to bacteria motility," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Prague, Czech Republic), pp. 5788–5791, May 2011.

[50] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *Journal of Theoretical Biology*, vol. 218, pp. 1–11, September 2002.

[51] S. Y. Tu and A. H. Sayed, "Effective information flow over mobile adaptive networks," in *Proc. 3rd International Workshop on Cognitive Information Processing (CIP)*, (Baiona, Spain), pp. 1–6, May 2012.

[52] S. Y. Tu and A. H. Sayed, "Mobile adaptive networks with self-organization abilities," in *Proc. 7th International Symposium on Wireless Communication Systems*, (York, UK), pp. 379–383, September 2010.

[53] S.-Y. Tu and A. H. Sayed, "Foraging behavior of fish schools via diffusion adaptation," in *Proc. 2nd International Workshop on Cognitive Information Processing*, (Elba, Italy), pp. 63–68, June 2010.

[54] S. Y. Tu and A. H. Sayed, "On the influence of informed agents on learning and adaptation over networks," *IEEE Trans. Signal Processing*, vol. 61, pp. 1339–1356, March 2013.

[55] A. H. Sayed, S.-Y. Tu, and J. Chen, "Online learning and adaptation over networks: More information is not necessarily better," in *Proc. Inf. Theory Appl. Workshop (ITA)*, (San Diego, USA), pp. 1–8, February 2013.

[56] A. Rastegarnia, A. Khalili, and M. K. Islam, "A self-organizing diffusion mobile adaptive network for pursuing a target," *American Journal of Signal Processing*, vol. 6, no. 2, pp. 25–31, 2016.

[57] N. Bogdanovi, J. Plata-Chaves, and K. Berberidis, "Distributed incremental-based LMS for node-specific parameter estimation over adaptive networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Vancouver, BC, Canada), pp. 5425–5429, May 2013.

[58] J. Plata-Chaves, M. H. Bahari, M. Moonen, and A. Bertrand, "Unsupervised diffusion-based LMS for node-specific parameter estimation over wireless sensor networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Shanghai, China), pp. 4159–4163, March 2016.

[59] Y. Wang, W. P. Tay, and W. Hu, "Multitask diffusion LMS with optimized inter-cluster cooperation," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, (Palma de Mallorca, Spain), pp. 1–5, June 2016.

[60] Y. Wang, W. P. Tay, and W. Hu, "A multitask diffusion strategy with optimized inter-cluster cooperation," *IEEE Journal of Selected Topics in Signal Processing*, pp. 504–517, April 2017.

[61] L. Jacob, F. Bach, and J.-P. Vert, "Clustered multi-task learning: A convex formulation," in *Proc. Neural Inform. Processing Systems (NIPS)*, (Vancouver, Canada), pp. 1–8, December 2008.

[62] S. Y. Tu and A. H. Sayed, "Distributed decision-making over adaptive networks," *IEEE Trans. Signal Processing*, vol. 62, pp. 1054–1069, March 2014.

[63] J. Chen, C. Richard, and A. H. Sayed, "Multitask diffusion adaptation over networks," *IEEE Trans. Signal Processing*, vol. 62, pp. 4129–4144, August 2014.

[64] X. Zhao and A. H. Sayed, "Clustering via diffusion adaptation over networks," in *Proc. International Workshop on Cognitive Inform. Processing (CIP)*, (Baiona, Spain), pp. 1–6, May 2012.

[65] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS over multitask networks," *IEEE Trans. Signal Processing*, vol. 63, pp. 2733–2748, June 2015.

[66] J. Chen, C. Richard, and A. H. Sayed, "Adaptive clustering for multitask diffusion network," in *Proc. 23rd European Signal Processing Conference (EUSIPCO)*, (Nice, France), pp. 200–204, September 2015.

[67] X. Zhao and A. H. Sayed, "Distributed clustering and learning over networks," *IEEE Trans. Signal Processing*, vol. 63, pp. 3285–3300, July 2015.

[68] J. Chen, C. Richard, and A. H. Sayed, "Diffusion LMS for clustered multitask networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (Florence, Italy), pp. 5487–5491, May 2014.

[69] J. Chen and C. Richard, "Performance analysis of diffusion LMS in multitask networks," in *Proc. 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, (St. Martin, France), pp. 137–140, December 2013.

[70] S. Y. Tu and A. H. Sayed, "Adaptive decision-making over complex networks," in *Proc. Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, USA), pp. 525–530, November 2012.

[71] M. Z. Lin, M. N. Murthi, and K. Premaratne, "Mobile adaptive networks for pursuing multiple targets," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, (South Brisbane, QLD), pp. 3217–3221, April 2015.

[72] J. Liu, M. Chu, and J. E. Reich, "Multitarget tracking in distributed sensor networks," *IEEE Signal Processing Magazine*, vol. 24, pp. 36–46, May 2007.

[73] X. Zhang, "Adaptive control and reconfiguration of mobile wireless sensor networks for dynamic multi-target tracking," *IEEE Trans. Autom. Control*, vol. 56, pp. 2429–2444, October 2011.

[74] O. Knill, *Probability Theory and Stochastic Processes with Applications*. Overseas Press India Private Limited, 2009.

[75] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Higher Education, 2002.

[76] J. Chen and A. H. Sayed, "On the probability distribution of distributed optimization strategies," in *Proc. Global Conference on Signal and Information Processing (GlobalSIP)*, (Austin, USA), pp. 555–558, December 2013.

[77] X. Zhao and A. H. Sayed, "Probability distribution of steady-state errors and adaptation over networks," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, (Nice, France), pp. 253–256, June 2011.

[78] J. Sacks, "Asymptotic distribution of stochastic approximation procedures," *The Annals of Mathematical Statistics*, vol. 29, pp. 373–405, June 1958.

[79] M. B. Nevel'son and R. Z. Khas' minskii, *Stochastic approximation and recursive estimation*. American Mathematical Society Providence, 1973.

[80] B. Fristedt and L. Gray, *A Modern Approach to Probability Theory*. Boston, MA: Birkhauser, 1997.

[81] P. Li, T. J. Hastie, and K. W. Church, "Nonlinear estimators and tail bounds for dimension reduction in ℓ_1 using Cauchy random projections," *J. Mach. Learn. Res.*, vol. 8, pp. 2497–2532, October 2007.

[82] M. Chiani, D. Dardari, and M. K. Simon, "New exponential bounds and approximations for the computation of error probability in fading channels," *IEEE Trans. Wireless Communications*, vol. 2, pp. 840–845, July 2003.

[83] N. R. Yousef and A. H. Sayed, "A unified approach to the steady-state and tracking analyses of adaptive filters," *IEEE Trans. Signal Processing*, vol. 49, pp. 314–324, February 2001.

[84] C. M. Grinstead and J. L. Snell, *Introduction to Probability.* AMS, 2003.

Curriculum Vitae

Name:	Sahar Khawatmi
Date of birth:	15 March 1986
Place of birth:	Bielefeld, Germany
Family status:	Married, one child

Education

2004–2008	Bachelor of Science at Computer Engineering, Electrical and Electronic Engineering, Aleppo University.
2003	High School Degree (Abitur) at Rajab Karmo High School, Aleppo, Syria.

Work experience

since 2011	Research Associate at Signal Processing Group, Technische Universität Darmstadt.
2009–2011	Research Assistant at Department of Computer Engineering, Aleppo University.

Erklärung laut §9 der Promotionsordnung

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Hereby I declare that I authored the present PhD-thesis alone and exclusively under the use of the literature denoted. The present PhD-thesis has by now not been used in any exam.

Darmstadt, 08. Mai 2017,