UNIVERSITA' DELLA CALABRIA

Dipartimento di Mathematica e Informatica

Dottorato di Ricerca in

Mathematica e Informatica

CICLO

XXXII

Distributed Optimization Over Large-Scale Systems for Big Data Analytics

Coordinatore: **Ch.mo Prof. N. Leone**

Firma _____

Supervisore/Tutor:

Dottorando: Dott./ssa (Reza Shahbazian)

Abstract

A large-scale system is defined as one that supports multiple, simultaneous users who access the core functionality through some network. Nowadays, enormous amount of data is continually generated at unprecedented and ever-increasing scales. Large-scale data sets are collected and studied in numerous domains, from engineering sciences to social networks, commerce, bimolecular research, and security. Big Data is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage, and process with acceptable latency. Usually, Big Data has one or more of the characteristics including high volume, high velocity, or high variety. Big Data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy and data source. Generally, Big Data comes from sensors, devices, video or audio, networks, log files, transactional applications, web, and social media, in a very large-scale. Big Data is impossible to analyze by using traditional central methods and therefore, new distributed models and algorithms are needed to process the data.

In this thesis, we focus on optimization algorithms for Big Data application. We review some of the recent machine learning, convex and non-convex, heuristic and stochastic optimization techniques and available tools applied to Big Data. We also propose a new distributed and decentralized stochastic algorithm for Big Data analytics. Our proposed algorithm is fully distributed to decide large-scale networks and data sets. The proposed method is scalable to any network configuration, is near real-time (in each iteration, a solution is provided although it might not be the optimum one) and more critical, robust to any missing data or communication failures. We evaluate the proposed method by a practical example and simulations on cognitive radio networks. Simulation results confirmed that the proposed method is efficient in terms of accuracy and robustness.

We assume that the distributed data-sources should be capable of processing their data and communicate with neighbor sources to find the network objective as an optimal decision. Some challenges are introduced by new technologies such as 5G or high-speed wireless data transfer, including imperfect communications that damage the data. We propose an optimal algorithm that uses optimal weighting to combine the shared data coming from neighbors. This optimal weight improves the performance of the decision-making algorithm in terms of error and convergence rate. We evaluate the performance of the proposed algorithm mathematically and introduce the step-sized conditions that guaranteed the convergence of the proposed algorithm. We use computer simulations to evaluate the network error. We prove that in a network diagram with ten data-sources, the network performance of the proposed algorithm outperforms some of the known optimal solutions such as Metropolis and adaptive combination.

Keywords: Optimization, Big Data, Large-Scale, Distributed, Optimal Weight.

Acknowledgments

My deepest gratitude goes to my advisers, Prof. Lucio Grandinetti, and Prof. Francesca Guerriero, for their continuous support and inspiration throughout my Ph.D. Program. Without their guidance and constructive advice, this thesis would never be created.

I would also like to express my sincere appreciation to my lovely wife, Leili who helped me throughout this thesis.

Preface & Contributions of the Author

The research presented in this dissertation was carried out at the Department of Mathematics and Computer Science, the University of Calabria from Sept. 2016 to Sept. 2019. This dissertation is the result of my work and created under the supervision of Prof. Lucio Grandinetti, and Prof. Francesca Guerriero from the Unical. The research has led to one journal, several conference publications and one book chapter.

The related contributions are listed below:

• Journal Papers

1. **Reza Shahbazian** and Francesco Guerriero, "Optimized Distributed Large-Scale Analytics Over Decentralized Data Sources with Imperfect Communication", Accepted, Journal of Supercomputing (2020) doi:10.1007/s11227-019-03129-5.

• Conference Papers

2. **Reza Shahbazian**, Lucio Grandinetti, Francesca Guerriero. "A New Distributed and Decentralized Stochastic Optimization Algorithm with Applications in Big Data Analytics." In International Conference on Machine Learning, Optimization, and Data Science, pp. 77-91. Springer, Cham, 2018.
3. **Reza Shahbazian**, Francesca Guerriero. "where optimization meets Big Data: A survey" HPC 2018, advances in parallel computing Series, Volume 34: Future Trends of HPC in a Disruptive Scenario, pp. 22-33, 2019. DOI:10.3233/APC190004
4. **Reza Shahbazian**, Francesca Guerriero. "Optimal Weight Design for Decentralized Consensus-based Large-Scale Data Analytics over Internet of Things." In International Congress on High-Performance Computing and Big Data Analysis (TopHPC), pp. 279-288. Springer, Cham, 2019.

• Book Chapter

5. **Reza Shahbazian**, Francesca Guerriero, and Seyed Ali Ghorashi. "Cooperative Ranging-Based Detection and Localization: Centralized and Distributed Optimization Methods." Cooperative Localization and Navigation: Theory, Research, and Practice (2019): 301.

Contents

1 Chapter 1 9

1.1 Introduction 10

1.2 Basics of Optimization 10

1.3 Big Data Analytics 11

1.3.1 Big Data History 14

1.4 Big Data and Optimization 15

1.5 Thesis objectives 16

1.6 Thesis Organization 17

2 Chapter 2 19

2.1 Introduction 20

2.2 Big Data Tools 20

2.3 Big Data optimization 22

2.3.1 Machine Learning for Big Data Analytics 22

2.3.2 Classic Optimization for Big Data Analytics 25

2.3.3 Heuristic and Evolutionary Optimization for Big Data 30

3 Chapter 3 32

3.1 Introduction 33

3.1.1 System Model 34

3.2 Proposed Method 37

3.2.1 Computational Complexity 38

3.3 Evaluation Results 39

3.3.1 Cognitive Networks 39

3.3.2 Conclusion 42

4 Chapter 4 43

4.1 Introduction 44

4.1.1 System Model 46

4.1.2 Optimal-Weight Distributed Algorithms 47
4.1.3 Effect of Imperfect Communication 48
4.2 Proposed Optimal Weight Design 50
4.2.1 Convergence Analysis 54
4.2.2 Evaluation Results 56
4.3 Conclusion 57
5 Chapter 5 59
5.1 Summary and Conclusion 60
5.2 Future Works 60
6 Appendix 68
6.1 Simulation Codes 69
References ……64

List of Figures

Figure 1-1 characteristics of Big Data including value, veracity, viscosity, virality, and visualization 11

Figure 1-2 The characteristics of Big Data from variety, volume, velocity, value, and veracity . 13

Figure 2-1 Partitioned optimization problem over a path graph of N=4 nodes for the randomized primal distributed algorithm for partitioned Big Data 26

Figure 2-2 Relationships among ADMM based algorithms for Big Data optimization..... 28

Figure 3-1 Different Characteristics of optimization algorithms for Big Data analytics 34

Figure 3-2 Instruction of performed tasks in distributed spectrum sensing..... 40

Figure 3-3 Performance of the distributed spectrum sensing when the communication link is ideal 40

Figure 3-4 Performance of the distributed spectrum sensing when the communication link fails with the probability of 0.4..... 41

Figure 4-1 Sample diagram of the sensory network in-which the data is transferred to cloud/Edge data-sources that can communicate. The network should make a decision or insight from such Big Data in a distributed manner. 45

Figure 4-2 The diagram of the network used for evaluation with 10 decentralized data-sources 56

Figure 4-3 Normalized network error versus the number of iterations for proposed method, Metropolis [55] and optimal combination method [56]..... 57

List of Acronyms

BDA	Big Data Analytics
KKT	Karush-Kuhn-Tucker
RFID	Radio-Frequency Identification
WSN	Wireless Sensor Network
HDFS	Hadoop Distributed File System
SDP	Semidefinite Programming
SOCP	Second-Order Cone Programming
ADMM	Alternating Direction Method of Multipliers
ITS	Intelligent Transportation Systems
BSUM	Block successive Upper-Bound Minimization
BCD	Block Coordinate Descent
CCCP	Convex Concave Procedure
BCPG	Block Coordinate Proximal gradient
NMF	Nonnegative Matrix Factorization
EM	Expectation Maximization

1 Chapter 1

Introduction

1.1 Introduction

In this chapter, we introduce the basics of optimization and its application on large-scale systems. We further present a description of Big Data, its history, definition and the relation with optimization techniques.

1.2 Basics of Optimization

A large-scale system is defined as one that supports multiple, simultaneous users who access the core functionality through some network. The term “Optimization” comes from the same root as “optimal,” which means best. When optimizing something, we are “making it best.” Mathematical Optimization is a branch of applied mathematics, which is useful in many different fields. The fundamental optimization problem consists of the objective function $f(x)$, which we are trying to maximize or minimize and variables x_1, x_2, x_3, which are the things we can control. They are abbreviated x_n to refer to individuals or to refer to them as a group. The general mathematical formulation of an optimization problem is presented as follows [1]:

$$\begin{aligned} &\underset{x}{\text{Min}}\ f(x) \\ s.t.\ &h(x) = 0, \\ &g(x) \le 0. \end{aligned} \tag{1-1}$$

Where x is a vector, including the n decision variables, $f(.)$ is the objective function of the optimization problem. It maps values of the decision vector x to a real value representing the desirability of this solution to the decision-maker. Typically, the objective function represents a cost in minimizing problems or a benefit in maximizing ones. $h(.)$ Also, $g(.)$ are vector-valued functions of the decision vector x. They define m equality and l inequality constraints.

Let's consider the general optimization formulation and suppose that $f(.)$ and $g(.)$ are continuously differentiable and convex, $h(.)$ is affine. Furthermore, we assume that a constraint qualification holds. For example, we may require that $g(.)$ be affine (linearity constraint qualification). Another standard constraint qualification requires linear independence of the gradients of active inequality constraint and equality constraints [1].

We can define the Lagrangian function for the optimization problem as follows [1]:

$$\begin{aligned} \nabla_x f(x) + \lambda^T \nabla_x h(x) + \mu^T \nabla_x g(x) &= 0, \\ h(x) &= 0, \\ g(x) &\le 0, \\ \mu &\ge 0, \\ \mu^T g(x) &= 0. \end{aligned} \tag{1-2}$$

Where ∇ is the gradient operator, λ, μ are the Lagrange multipliers and constraint qualifications are needed for ensuring that KKT (Karush-Kuhn-Tucker) conditions.

1.3 Big Data Analytics

Big Data is a term applied to data sets whose size or type is beyond the ability of traditional relational databases to capture, manage, and process with acceptable latency. Usually, Big Data has one or more of the characteristics, including high volume, high velocity, or high variety, also named 3-Vs. Nowadays, more V words are mentioned as characteristics of Big Data including value, veracity, viscosity, virality, and visualization, [2], as depicted in Figure 1-1.

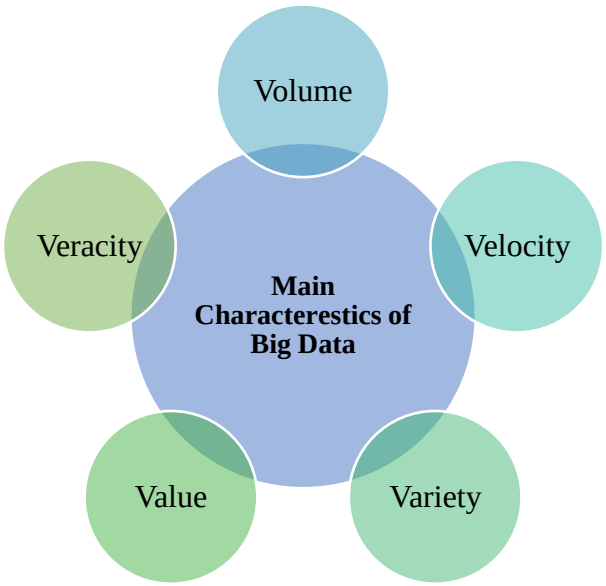

Figure 1-1 characteristics of Big Data including value, veracity, viscosity, virality, and visualization

Big Data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy, and data source. Current usage of the term

Big Data tends to refer to the use of predictive analytics, user behavior analytics, or specific other advanced data analytics methods that extract value from data, and seldom to a particular size of data set. Data sets overgrow, in part because they are increasingly gathered by cheap and numerous information-sensing Internet of things devices such as mobile devices, aerial (remote sensing), software logs, cameras, microphones, radio-frequency identification (RFID) readers and wireless sensor networks or sensors on a production line and communications in a social network [2].

Relational database management systems, desktop statistics, and software packages used to visualize data often have difficulty handling Big Data [2]. The work may require "massively parallel software running on tens, hundreds, or even thousands of servers." Big Data philosophy encompasses unstructured, semi-structured, and structured data; however, the main focus is on unstructured data [3]. The characteristics of 5 Vs on Big Data is depicted in Figure 1-2 and explained as follows:

Volume: The quantity of generated and stored data. The size of the data determines the value and potential insight, and whether it can be considered as Big Data or not.

Variety: The type and nature of the data. This helps people who analyze it to effectively use the resulting insight. Big Data draws from text, images, audio, video; plus it completes missing pieces through data fusion.

Velocity: The speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development. Big Data is often available in real-time. Compared to small data, Big Data is produced more continually. Two kinds of velocity related to Big Data are the frequency of generation and the frequency of handling, recording, and publishing.

Veracity: Refers to the data quality and the data value. The data quality of captured data can vary greatly, affecting the accurate analysis.

Value: The utility that can be extracted from the data.

Figure 1-2 The characteristics of Big Data from variety, volume, velocity, value, and veracity

Big Data comes from sensors, devices, video or audio, networks, log files, transactional applications, web, and social media, in a very large-scale [2]. Currently, more than 2.7 zeta bytes of data exist in the digital universe; Twitter processes over 70 million tweets and business transactions on the Internet will reach 450 billion per day by the year 2020. By 2020, the amount of data just in China is expected to reach 8.6 zeta bytes. It is estimated that by 2020, there will be more than 30 billion connected devices.

Big Data requires new forms of processing for insight discovery and enhanced decision making in which optimization may play a critical role. It is a known fact that enormous potential value is hidden in Big Data, and meaningful application of it will change human society, dramatically. Big

Data enables tremendous potential in terms of business value in a variety of fields such as health-care, biology, transportation, advertising, energy management, and financial services. A few years ago, Big Data was very new. However, we are facing new generations of Big Data. For instance, recently, multimedia Big Data is introduced that is a type of datasets, in which the data has more media types and higher volume than the typical Big Data [4]. Multimedia Big Data analysis requires more sophisticated algorithms and much more computing resources compared with existing Big Data systems [4].

Big Data could be viewed from different aspects. Some researchers are focused on the application of Big Data in a specific field. Big Data has applications in the new generation of mobile cellular systems, including 4.9G and 5G, Smart Grid, or even intelligent transportation systems (ITS) [3-8]. Researchers also develop tools and frameworks that could be used for Big Data analytics (BDA). In continue, we provide a short brief on Big Data analytics tools.

1.3.1 Big Data History

Big Data repositories have existed in many forms, often built by corporations with a particular need. Commercial vendors historically offered parallel database management systems for Big Data beginning in the 1990s. For many years, WinterCorp published the most significant database report. Teradata Corporation, in 1984, marketed the parallel processing DBC 1012 system. Teradata systems were the first to store and analyze one terabyte of data in 1992. Hard disk drives were 2.5 GB in 1991, so the definition of Big Data continuously evolves according to Kryder's Law. Teradata installed the first petabyte class RDBMS based system in 2007. As of 2017, there are a few dozen petabyte class Teradata relational databases installed, the largest of which exceeds 50 PB. Systems up until 2008 were 100% structured relational data. Since then, Teradata has added unstructured data types, including XML, JSON, and Avro.

In 2000, Seisint Inc. (now LexisNexis Risk Solutions) developed a C++-based distributed platform for data processing and querying known as the HPCC Systems platform. This system automatically partitions, distributes, stores, and delivers structured, semi-structured, and unstructured data across multiple commodity servers. Users can write data processing pipelines and queries in a declarative dataflow programming language called ECL. Data analysts working in ECL are not required to define data schemas upfront and can instead focus on the particular problem at hand, reshaping data in the best possible manner as they develop the solution.

In 2004, LexisNexis acquired Seisint Inc. and its high-speed parallel processing platform and successfully utilized this platform to integrate the data systems of Choicepoint Inc. when they

acquired that company in 2008. In 2011, the HPCC systems platform was open-sourced under the Apache v2.0 License.

CERN and other physics experiments have collected Big Data sets for many decades, usually analyzed via high-performance computing (supercomputers) rather than the commodity map-reduce architectures traditionally meant by the current "Big Data" movement [9].

In 2004, Google published a paper on a process called MapReduce. The MapReduce concept provides a parallel processing model, and an associated implementation was released to process vast amounts of data. With MapReduce, queries are split and distributed across parallel nodes and processed in parallel (the Map step). The results are then gathered and delivered (the Reduce step). The framework was very successful, so others wanted to replicate the algorithm. Therefore, an implementation of the MapReduce framework was adopted by an Apache open-source project named Hadoop. Apache Spark was developed in 2012 in response to limitations in the MapReduce paradigm, as it adds the ability to set up many operations (not just map followed by reducing).

Big Data analytics for manufacturing applications is marketed as a "5C architecture" (connection, conversion, cyber, cognition, and configuration). Factory work and Cyber-physical systems may have an extended "6C system":

- Connection (sensor and networks)
- Cloud (computing and data on demand)
- Cyber (model and memory)
- Content/context (meaning and correlation)
- Community (sharing and collaboration)
- Customization (personalization and value)

Techniques for analyzing data, such as A/B testing, machine learning, and natural languages processing Big Data technologies, like business intelligence, cloud computing and databases visualization, such as charts, graphs and other displays of the data multidimensional Big Data can also be represented as data cubes or, mathematically, tensors. Array Database Systems have set out to provide storage and high-level query support on this data type [9].

1.4 Big Data and Optimization

Big Data is impossible to analyze by using traditional central methods [8], and therefore, distributed processing with parallelization is needed. Data analytics often must be performed in

real-time or near real-time. Gaining an answer to the analysis demands on almost real-time is almost preferred to a precise decision but in a timely manner. Optimization algorithms for Big Data aim to reduce the computational, storage, and communications challenges [8]. The data and parameter sizes of Big Data optimization problems are too large to process locally, and since the Big Data models are inexact, optimization algorithms no longer need to find the high accuracy solutions. It means that, unlike usual optimization problems that try to provide an exact model and precise solution, Big Data analytics expects to reach an insight or decision even if it is not the accurate answer to the problem [8].

In a straightforward explanation, Big Data optimization methods try to partition the data so that it is feasible to process, mostly in a centralized manner. Although many research articles are published in this era, there is still a big gap between practice and theory, especially considering the needs for scalability, robustness, and characteristics of Big Data. It seems that distributed optimization algorithms are a promising solution to fill this gap, although there still is a long road to go. In the next chapter, we review the recent advances in Big Data optimization and address the research articles related to Big Data optimization.

1.5 Thesis objectives

Large scale datasets can not be processed using traditional methods and algorithms. The researchers and companies have introduced new tools and techniques for the analysis of such large scale data sets.

In this thesis, our goal is to present new approaches to be used in large scale dataset processing, including Big Data, which is more focused on applicable mathematical aspects rather than tools. We propose a distributed optimization algorithm, including consensus and diffusion adaptation. We also consider some new practical issues in future data analysis that is using next-generation mobile networks. We model the effect of communication failure in distributed optimization algorithms that we earlier proposed for Big Data analytics. In general, we have tried to answer the following questions in this research.

- What is the definition and characteristics of Big Data?
- What tools and frameworks are already used for Big Data analytics?
- What is the relationship between mathematical optimization techniques and Big Data analytics?

- Is it possible to propose new algorithms for Big Data analytics that are scalable, perform near real-time, and achieve near-optimal solutions?
- What are the effects of new technologies, such as next-generation mobile networks (5G) on Big Data analytics?

We also liked to review and contribute to classical distributed optimization algorithms for Big Data analytics, which was not successful during this research period.

To answer the above questions, we have performed a review on state of the art and recent articles followed by modeling the Big Data analytics as a mathematical optimization problem. We further propose a diffusion-based algorithm with a gradient-based iterative solution that satisfies scalability, real-time, and decentralized needs. We also mathematically model the effect of imperfect communication for next-generation mobile networks used in Big Data concept and propose an adaptive weighting approach to overcome the issue. We provide some examples in different scenarios, including cognitive radio and the Internet of things, to explain the proposed methods more clearly.

1.6 Thesis Organization

In chapter 2 of this thesis, we review some of the recent and relevant optimization algorithms used for Big Data analysis, including machine learning algorithms, classic optimization algorithms, and heuristic and evolutionary optimization algorithms. We also review some of the most useful tools in Big Data analysis and categorize them into several tables.

In chapter 3, we show that distributed algorithms are a solution to the decentralized nature of Big Data algorithms. We present the basics of stochastic distributed decision-making algorithms and propose a fully distributed one that is based on diffusion adaptation. The proposed algorithm uses the collaboration of neighbor nodes to make a global decision while the sources make local decisions. We evaluate the proposed algorithm on cognitive radio networks in which secondary users use energy detector sensors to sense the environment. With some computer simulations, we show that the proposed algorithms perform well, are scalable and robust to communication failures.

In chapter 4, we consider some practical issues, including the effect of imperfect communications on distributed analysis algorithms. We model this effect as an optimization problem and solve the problem to gain the optimal weights that each node assigns to the information received from

neighbor nodes. We compare the proposed optimal weighing algorithm with some of the earlier methods. Evaluation results confirm that the proposed method performs well in terms of error and convergence rate.

2 Chapter 2

Where Big Data meets optimization

2.1 Introduction

In this section, we provide an overview of optimization methods used for Big Data Analytics (BDA) like first-order methods, randomization, heuristic, evolutionary, and convex algorithms.

Big Data is a common topic, and many researchers in various fields of study, including convex optimization and machine learning, have contributed to the literature. The essential ingredient for every smart and intelligent system is data. More energetic systems acquire more data to make an efficient decision that leads to large-scale data sets. This data could be generated from many sensors in smartphones, physical sensors attached to cyber-physical systems, many objects in the Internet of Things (IoT) platforms and smart cities. This data may further be transferred to a center using new technologies such as 5G. Therefore, data gathering is the first challenge of Big Data Era. Other problems may include data storage and data processing. Many types of research are focused on the adaptation of existing technologies or inventing new ones to store Big Data. However, many researchers believe that the main challenge is still finding efficient and optimal solutions to process the data in the appropriate time by considering the Big Data challenges [3-4].

Nowadays, new generations of Big Data, such as multimedia Big Data, are introduced [4]. In this new generation, data has more media types and higher volume than the typical Big Data. As a result, there is an increasing demand to develop new models and tools to analyze large-scale complex networks in-which the optimization plays a vital role [4]. Such optimized analysis models and algorithms should be designed so that there is no need for a central authority, but a decentralized architecture is required, and it applies to any network structure.

2.2 Big Data Tools

Data analytics involves various tools such as those from text analytics, business intelligence, data visualization, and statistical analysis. The tools used for Big Data analytics have a long story, and it is not possible to explain them all in a few lines. However, it is possible to highlight some important ones.

Hadoop and MapReduce are the most common tools for Big Data analytics [2]. Hadoop was initially developed by Cloudera while Google created the MapReduce. Hadoop provides a distributed file system and provides the required storage facilities in the form of Hadoop distributed file system (HDFS). MapReduce prepares the process of data by dividing it into a manageable size and distributing it to different nodes for analysis. Hadoop and MapReduce require

a cluster of computers for BDA. Most of the other available tools are built upon the features of Hadoop and MapReduce [2].

Hadoop and MapReduce are free. Besides these free tools, many companies have developed their frameworks. Microsoft provides a suite of Big Data services named Azure. Azure Data Lake provides all the capabilities for the interactive handling of Big Data and includes services like HBase, and Storm for integration with Hadoop. IBM provides business analytics and optimization solutions for Big Data analytics. IBM BigInsights is a Hadoop-based tool for enterprise requirements, and IBM Streams is a platform for real-time analytic processing. SAND Analytic Platform is an analytic database platform that has the capability for massively parallel processing. SAP offers analytical platforms for business intelligence and data warehousing [10].

SAS Supports cloud-based analytics while the SAS data integration studio provides support for Hadoop. Tableau, with its rich visualization features and ease of use capabilities, enables non-expert users to explore and exploit enterprise data. Pentaho is another tool that proposes rich visualization and analytics capabilities. Oracle provides Big Data and cloud services through Oracle Big Data cloud service, Oracle Big Data SQL cloud service, Oracle database cloud service, and Oracle database Exadata cloud service [10].

Oracle advanced analytics includes Oracle data miner, Oracle R advanced analytics for Hadoop and Oracle Big Data discovery, along with connectors and interfaces for SQL and R [11]. Besides these tools as platforms, some tools are also available for database and data processing. The categorization of some known tools used for Big Data analytics is presented in Table 1.

Table 1. Categorization of some known tools used for Big Data Analytics

Platform	Local	Hadoop, MapReduce, Cloudera, Hortonworks, BigInsights
	Cloud	AWS, Google Compute Engine, Azure
Data Base	SQL	Greenplum, Aster data, Vertica
	NO-SQL	HBase, Cassandra, MongoDB, Redis
	New-SQL	Spanner, Megastore, FI
Data Processing	Batch	MapReduce, Dryad
	Stream	Storm, S4, Kafka
Data Warehouse		Hive, HadoopDB, Hadopt
Query Language		HiveQL, Pig Latin, DryadLINQ, MRQL, SCOPE
Statistics and Machine Learning		Mahout, Weka, R

For data storing, we may also categorize the tools based on the storage method, including column, document, graph, and key-value, as presented in Table 2.

Table 2. Categorization of data storage tools used for Big Data Analytics

Data Storage	**Column**	Cassandra, HBase, Hypertable
	Document	MongoDB, JSON ODM, CouchDB
	Graph	Neo4j, ArangoDB, Bigdata
	Key-value	DynamoDB, Azure Table Storage, Cassandra, PNUTS, Berkeley DB

2.3 Big Data optimization

In this section, we assume that massive amounts of data already exists (without considering the source) and review recent advances and state of the art, including convex, non-convex, heuristic, evolutionary, Game-theory and machine learning-based methods.

2.3.1 Machine Learning for Big Data Analytics

In the literature, machine learning plays an essential role in Big Data analytics (BDA). In continue, we first review the general application of machine learning for BDA. Traditional machine learning approaches developed based on this assumption; the data set will entirely fit into memory that no longer holds in the Big Data context. This broken assumption, together with the Big Data characteristics, create obstacles to use traditional techniques [3].

The two main categories of learning tasks are: supervised and unsupervised. In supervised learning, inputs and desired outputs (labels) are known, and the system learns to map inputs to outputs. In unsupervised learning, desired outputs are not known, and the system itself discovers the structure within the data. Examples of supervised learning include classification and regression. In classification, the outputs take discrete values while in regression, the outputs are continuous. Some well-known classification algorithms are a k-nearest neighbor, logistic regression, and support vector machine (SVM). On the other hand, some of the famous regression includes support vector regression (SVR), linear regression, and polynomial regression. Some algorithms, such as neural networks can be used for both classification and regression [3].

Unsupervised learning includes clustering objects that are grouped based on similarity criteria. Predictive analytics use past data to predict the future; numerous algorithms including SVR, neural networks, and Naive Bayes [3].

In the machine learning context, the data size can be defined either vertically or horizontally. The vertical definition deals with the number of records in a dataset, while the horizontal definition considers the number of features or attributes data contains. Volume is relative to the type of data: a smaller number of very complex data points may be regarded as equivalent to a larger quantity of simple data [3]. One of the main challenges encountered in computations with Big Data comes from the simple principle that scale or volume adds computational complexity. Consequently, as the scale becomes large, even simple operations become costly.

For example, the standard SVM algorithm has a training time complexity $O\left(n^3\right)$ and a space complexity of $O\left(n^2\right)$, where n is the number of training samples. Therefore, increasing the size n, the SVM algorithm may become computationally infeasible on enormous datasets [3].

The time complexity of maximum likelihood-based algorithms, including principal component analysis, logistic regression, locally weighted linear regression and Gaussian discriminative analysis, is $O\left(nm^2+m^3\right)$ where n are the number of samples and m the number of features. For all these algorithms, the time needed to perform the computations will increase exponentially with increasing data size and may even render the algorithms unusable for enormous datasets. Therefore, the computational complexity of algorithms is another reason why existing machine learning and data analysis algorithms can not necessarily be used for Big Data analytics.

On the other hand, some existing machine learning algorithms are inherently parallel and can be adapted to the MapReduce paradigm, whereas others are difficult to take advantage of large numbers of computing nodes (parallel computing) [3]. In summary, traditional methods should be revised by considering new constraints.

When it comes to performance, optimization plays a pivotal role. Based on different perspectives available of Big Data, it is evident that the optimization can be applied to a variety of fields including database, query, search engine or processing. In [7] authors study Big Data-based self-optimization networking in next-generation mobile networks. Their goal is to maximize the network efficiency and increase the quality of services provided to microcell and femtocell users, considering the limited resources. They use Big Data techniques to decide to adjust network parameters in a distributed intelligence manner.

Authors in [8] propose distributed optimization storage and statistical system model-based. In [12], the authors propose an optimization algorithm based on the spatial and temporal data compression for wireless sensor networks in the underground tunnel environment. They introduce spatial and temporal correlation functions for data compression and data recovery. In [13], the authors study

means of integrating Big Data analytics with network optimization to improve the user quality of experience. They propose a framework of Big Data-driven mobile network optimization.

Authors in [14] explain how functions can serve to model the various data structures used to represent large data sets. They present details of four functions modeling tabular data, graph structures, cached, and split data. In [15], the authors present an optimization framework based on the Fireworks algorithm for Big Data optimization problems. Their proposed structure is composed of a single objective Fireworks algorithm and a multi-objective Fireworks algorithm to solve the significant optimization of signals problem "Big-OPT."

In [16] the query optimization process for performance up-gradation of cloud transactions and compare different strategies used in the optimization domain of distributed cloud databases are addressed. Authors in [17] propose an optimization algorithm for the massive data communication between the weather research and forecasting model and coupler version 7 in the Chinese Academy of Sciences-Earth System Model (CAS-ESM). Their optimization strategy is to transmit data from a small packet into a larger packet.

Recently, deep learning has absorbed the widespread adoption of enterprises to gain greater insight from the analytics. Although artificial intelligence (AI) was first conceived in the late 1950s, the recent jump into learning methods such as deep learning is fueled by the latest advancements in hardware industries. The combination of deep learning and Big Data appears in many research topics. For instance, in natural language processing techniques, the typed text requires lots of data to produce the best results possible (currently around 4% error). Different accents (in spoken language processing) are also needed. Similarly, we may consider the IoT applications. For instance, in the oil industry, we have several sensors taking measurements of critical factors in each pipeline. This rich sequential data stream (Big Data) is significant for deep learning [18].

A recent exciting topic in this area includes parallel and distributed network training for Big Data. In [19] authors study deep neural networks (DNN) as is used in image recognition, object detection, classification and tracking, and speech and language processing applications. They try to argue the training cost in computation and time and describe how to enable parallel deep neural network training on the IBM Blue Gene/Q (BG/Q) computer system. They explore DNN training using the second-order optimization algorithm [19].

Authors in [20] investigate the problem of the compute-intensive process of training proposes to use variants of the stochastic gradient descent (SGD) algorithm as a solution. In summary, they present optimization techniques to improve the performance of the data-parallel synchronous SGD algorithm using the Torch framework. This area is almost new, and many quality research articles are expected to be published in the near future.

In another perspective, we may categorize the Big Data optimization algorithms into two categorize, namely classic and heuristic algorithms. Classic algorithms include convex, dual descent, alternating direction method of multipliers (ADMM), while heuristic algorithms include some evolutionary algorithms. Next, we will review some recent developments of the classic optimization algorithms that are designed for Big Data applications.

2.3.2 Classic Optimization for Big Data Analytics

One of the reasonable efforts goes back to 2014, where Cevher and et al. in [21] study the convex optimization for Big Data. Convex optimization is applicable for Big Data, where the data and parameter sizes of optimization problems are too large to process locally. The fundamentals of Big Data convex optimization could be presented as follows:

$$\underset{x}{Min}\left\{F(x) := f(x) + g(x) : x \in \Re^N\right\} \tag{2-1}$$

Where f and g are convex functions. In general, three types of algorithms are used for Big Data convex optimization, namely First-order methods, Randomization, and Parallel computation.

First-order methods obtain low or medium accuracy numerical solutions by using only first-order information from the objective, such as gradient estimates. These methods feature nearly dimension-independent convergence rates, they are theoretically robust to the approximations, and they typically rely on computational primitives that are ideal for distributed and parallel computation. Randomization techniques particularly stand out among others to enhance the scalability of first-order methods since their expected behavior is controllable. Key ideas include random. Partial updates of optimization variables, replacing the deterministic gradient and proximal calculations with cheap statistical estimators, and speeding up basic linear algebra routines via randomization [21].

It is possible to further augment first-order methods with approximations for increasing levels of scalability, from idealized synchronous parallel algorithms with centralized communications to enormously scalable asynchronous algorithms with decentralized communications. These algorithms are called parallel and distributed computation. The three concepts above complement each other to offer scalability benefits for Big Data optimization [21].

Not every problem in Big Data is convex [46-47]. In [22], a randomized primal distributed algorithm for partitioned non-convex problems is proposed. In this method, a network of N nodes is modeled as an undirected communication graph with an edge (i, j) that indicates the edge

between node i and node j. These nodes can exchange information, and the cost function is separable. They try to split the effective cost into a smooth part (by modeling local objectives) and non-smooth one being a regularization term or constraint. They shape the problem is presented in equation (2-2) [22].

$$\min_{x} \sum_{i=1}^{N} \left(f_i\left(x_{N_i}\right) + g_i\left(x_i\right) \right) \tag{2-2}$$

Where the node i knows only the functions f_i and g_i. The problem is called partitioned due to the structure of the functions f_i and g_i. In equation (2-2) the N_i shows the set of neighbor nodes to the node i (connected nodes in the graph with edges). A sample visualization of the problem and the partitioned mechanisms $N = 4$ depicted in Figure 2-1.

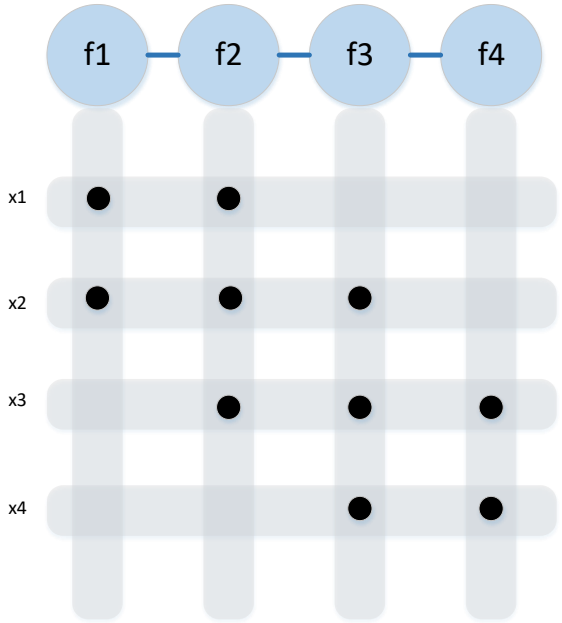

Figure 2-1 Partitioned optimization problem over a path graph of N=4 nodes for the randomized primal distributed algorithm for partitioned Big Data

In [23], authors consider the same problem and propose a distributed partitioned optimization via asynchronous dual decomposition. Similar to [22], they assume that the dimension of the decision variable depends on the network size, and cost function and constraints have a sparsity structure related to the communication graph.

In [24-25] authors consider Big Data optimization by using block communications. In [24], multi-agent large-scale optimization problems is studied wherein the cost function is composed of a smooth possibly non-convex utility and a difference of convex regularize. Authors assume that the

dimension of the optimization variables is so large that optimizing and transmitting the entire set of variables could cause un-affordable computation and communication overhead. The considered optimization problem can be written as follows [24].

$$\begin{aligned} &\underset{x}{Min} \quad U(x)=\sum_{i=1}^{N} f_i(x)+\sum_{l=1}^{B} r_l(x_l) \\ &S.t. \quad x=\left[x_1^T,\ldots,x_B^T\right]^T \\ &\quad x_l \in k_l, \qquad \forall l \in \{1,\ldots,B\} \end{aligned} \tag{2-3}$$

Where x is the vector of the optimization variables, partitioned in B a block, whole l^{th} block is denoted by x_l and f_i is a smooth possibly non-convex cost function of agent i and r_l is a difference of convex function commonly known by all the agents and k_l is a closed convex set. The term r_l usually plays the role of regulation. In [25] the problem is the same as [24] while the solution provided is based on an iterative algorithm called BLOCK-SONATA. Their evaluation results show that as the number of blocks increases, the performance improves.

Alternating direction method of multipliers (ADMM) is a distributed optimization method proposed in 2011. The methods ingredients are rather old and belong to the 60s and 70s. However, it is proven to show outstanding performance in different applications. In [26] authors propose a multi-block ADMM for Big Data optimization in modern communication networks. They consider a convex separable problem with a canonical form, as presented in equation (2-4).

$$\begin{aligned} &\underset{x_1,x_2,\ldots,x_N}{Min} \quad f(x)=f_1(x_1)+\cdots+f_i(x_N) \\ &S.t. \quad A_1x_1+\cdots+A_Nx_N=c, \\ &\quad x_i \in X_i, \qquad i=1,\ldots,\mathrm{N} \end{aligned} \tag{2-4}$$

For the solution of ADMM based optimization methods in [26], one of the situations presented in Figure 2-1, occurs. The convergent of solutions for of ADMM based optimization methods is presented in Figure 2-2.

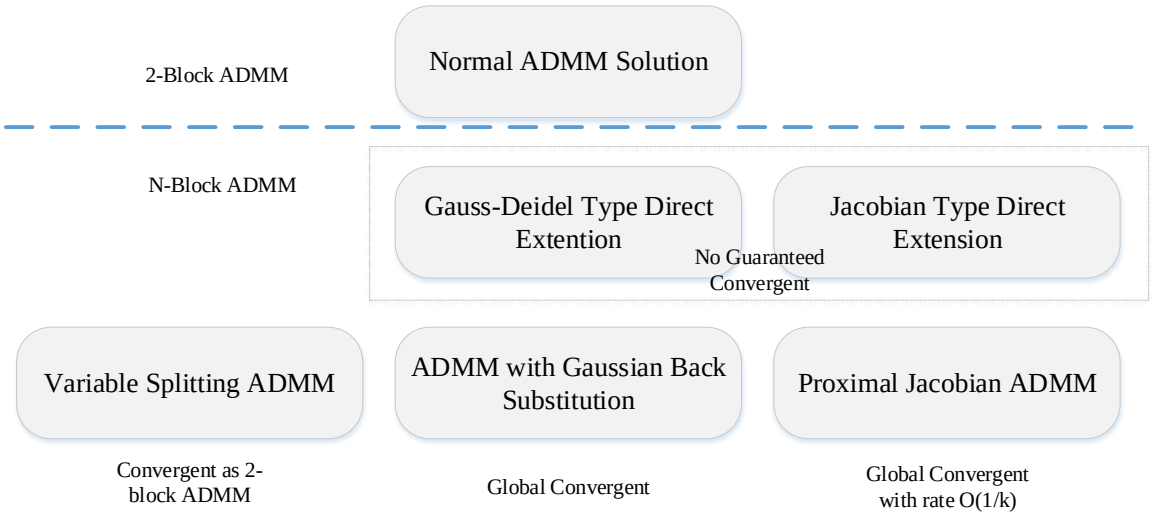

Figure 2-2 Relationships among ADMM based algorithms for Big Data optimization

In [27], parallel coordinate descent methods for Big Data optimization is proposed. The authors show that randomized (block) coordinate descent methods can be accelerated by parallelization when applied to the problem of minimizing the sum of a partially separable smooth convex function and a simple separable convex function. In their proposed method, when no degree of separability is present, there may be no speedup; in the best case, when the problem is separable, the speedup is equal to the number of processors.

In [28], the authors use the sub-gradient method for dual decomposition and propose an accelerated distributed optimization for the reconstruction of big sensory data. They show that the proposed approach converges in order $O\left(1/t^2\right)$. They use compressive sensing for sensory data. Another compressive sensing based optimization algorithms utilize the previous step information to update the intermediate variables while their proposed accelerated method it makes use of more past information by preserving more past information rather than only the last step. Therefore, the convergence speed is accelerated from $O\left(1/t\right)$ to $O\left(1/t^2\right)$.

Stochastic optimization for Big Data is studied in [29]. Authors propose a data-driven stochastic robust optimization named (DDSRO) for optimization under uncertainty leveraging labeled multi-class uncertainty data. They use machine learning methods, including the Dirichlet process mixture model and maximum likelihood estimation for uncertainty modeling. Their proposed algorithm is a two-stage stochastic programming approach to optimize the expected objective across different data classes followed by robust adaptive optimization as the inherent problem to ensure the

robustness of the solution. Their optimization problem is presented in equation (2-5) as follows [29].

$$\begin{aligned} &\min_{x} && \mathbf{c}^T\mathbf{x} + \mathbf{E}_{\sigma\in\Pi}\left[\bar{C}_\sigma(\mathbf{x})\right] \\ &S.t. && \mathbf{Ax} \geq \mathbf{d} \\ &\left\{ \begin{array}{ll} & \bar{C}_\sigma(\mathbf{x}) = \max_{u\in U_\sigma} \min_{y_\sigma} \mathbf{b}^T\mathbf{y}_\sigma \\ S.t. & \mathbf{Wy}_\sigma \geq \mathbf{h} - \mathbf{Tx} - \mathbf{Mu} \\ & \mathbf{y}_\sigma \in R_+^{n_3} \end{array} \right\}, \quad \forall\sigma\in\Pi \end{aligned} \tag{2-5}$$

where σ is an uncertain scenario that influences the uncertainty set and Π is the set of scenarios. The parameter $\mathbf{x}$ is the first stage decision made before the uncertainty $\mathbf{u}$ is realized, while $\mathbf{y}$ in the second stage decision, $\mathbf{c}^T\mathbf{x}$ and $\mathbf{b}^T\mathbf{y}$ are objective functions, $\mathbf{Ax} \geq \mathbf{d}$ and $\mathbf{Wy}_\sigma \geq \mathbf{h} - \mathbf{Tx} - \mathbf{Mu}$ are modeled constraints [29].

Low-rank modeling plays an essential role in signal processing and machine learning, with applications on Big Data analytics. Many high-dimensional data and interactions can be modeled as approximately in a low-dimensional subspace or manifold, possibly with additional structures applicable both in convex and non-convex approaches. Convex relaxations such as nuclear norm minimization often lead to statistically optimal procedures for estimating low-rank matrices, where first-order methods are developed to address the computational challenges. There is emerging evidence that correctly designed non-convex procedures, such as projected gradient descent, often provide globally optimal solutions with a much lower computational cost in many problems. In [30] authors' survey article gives a unified overview of these recent advances in low-rank matrix estimation from incomplete measurements. Attention is paid to rigorous characterization of the performance of these algorithms and to problems where the low-rank matrix has additional structural properties that require new algorithmic designs and theoretical analysis. Authors in [45] present an algorithmic framework for Big Data optimization, called the block Successive Upper Bound Minimization (BSUM). Their proposed BSUM includes methods such as the Block Coordinate Descent (BCD), the Convex Concave Procedure (CCCP), the Block Coordinate Proximal gradient (BCPG), the Nonnegative Matrix Factorization (NMF) and the Expectation-Maximization (EM).

2.3.3 Heuristic and Evolutionary Optimization for Big Data

There has been a growing interest in algorithms that are based on the principle of evolution [31-37]. A commonly accepted term refers to such techniques as evolutionary computation (EC) methods. The best-known algorithms in this class include genetic algorithms and evolutionary programming. Heuristic algorithms are designed to solve a problem in a faster and more efficient fashion than traditional methods by sacrificing optimality, accuracy, precision, or completeness for speed. Heuristic algorithms frequently used to solve NP-complete problems. Heuristic algorithms can produce a solution individually or be used to provide a good baseline and are supplemented with optimization algorithms. Heuristic algorithms are most often employed when approximate answers are satisfactory and exact solutions are necessarily computationally expensive. Here, we will review the research works from the heuristic and evolutionary optimizations algorithms that are focused on Big Data.

In [31], the authors focus on modeling and optimization of features selection in Big Data. They present a system architecture that selects features by using an artificial bee colony followed by a Kalman filter used in the Hadoop ecosystem for removal of noise. They also propose a complete four-tier architecture that aggregates the data, eliminate unnecessary data, and analyze the data by the Hadoop-based artificial bee colony algorithm. In [32], the authors study the differential evolution framework for Big Data optimization. They assume that a real-time Big Data problem is not known in advance. Therefore, they propose a general differential evolution framework in which the most suitable differential evolution algorithm for a problem on hand is adaptively configured. A local search is also employed to increase the exploitation capability of their proposed algorithm.

In [33] authors claim that multi-objective evolutionary algorithms (MOEAs) suffer from some difficulties when solving Big Data optimization problems with thousands of variables and propose a meta-heuristic firefly algorithm (FA) to solve the problem. In [34] an adaptive mutation operator is introduced to enhance the performance of the standard non-dominated sorting genetic algorithm, the third generation also called NSGA-III algorithm to overcome the high computational costs of solving Big Data optimization problems via traditional multi-objective evolutionary algorithms. jMetalSP [35] combines the multi-objective optimization features of the jMetal framework with the streaming facilities of the Apache Spark cluster computing system as software platforms to solve dynamic multi-objective Big Data optimization problems.

Genetic Optimization for Big Data analysis is presented in [36]. Authors in [36] consider opinion mining from unstructured textual documents and propose a method focused on minimum

preliminary requirements about the knowledge of the analyzed language. Their proposed method is built on artificial intelligence consisting of support vector machines classifier, Big Data analysis, and genetic algorithm optimization. To make available optimization for Big Data, they propose GA operators, which significantly accelerate conversion to accurate solutions.

In [37], authors try to study the known issues of evolutionary algorithms including scalability when dealing with Big Data problems. They propose a different framework that integrates a cooperative co-evolution method. They use the collaborative co-evolution method to split the big problem into subproblems to increase the efficiency of the solving process. The sub-problems are then solved using various heterogeneous memetic algorithms. Their proposed different framework adaptively assigns different operators, parameter values and a local search algorithm to efficiently explore and exploit the search space of the given problem instance.

Conclusion

Every day, massive amounts of data are collected by sensors, log files, networks, and smart devices, forming enormous volume, velocity, and variety. The data with one or more than of these characteristics is called Big Data. Processing Big Data enables smart decisions and insight. However, it is impossible to analyze the traditional central methods. Optimization algorithms for Big Data aim to reduce the computational, storage, and communications challenges. In this section, we provided an overview of optimization methods used for Big Data Analytics (BDA) like first-order methods, randomization, heuristic, evolutionary, and convex algorithms.

In summary, optimization algorithms used for Big Data analytics should perform in a distributed manner. They split the workload and process each part using parallel computing methods. The recent significant algorithms used for Big Data analytics include primal-dual, ADMM, randomized, and stochastic methods. All these methods require iterative solutions to decide on Big Data analytics.

3 Chapter 3

A new Distributed Stochastic Algorithm for decision making over large-scale datasets

3.1 Introduction

Data processing can be performed in a centralized or distributed manner [38]. Analysis of large data sets and Big Data seems infeasible by using central processing and storage units [48]. Considering the streaming data sources, learning must often be performed in real-time or near real-time [38]. Although centralized processing methods usually provide the optimal decision, considering the challenges faced by data storage in the cloud or any distributed file system [39], decentralized methods are still preferred [40]. Therefore, there is an urgent need for scalable methods, capable of efficient data processing, considering the storage, query, and communication challenges. In some cases, privacy and security concerns are critical and prevent accessing the full data. In these cases, only partial data or processed output (decision) might be transferred through communication interfaces.

As depicted in Figure 3-1, the characteristics of Big Data require an optimization algorithm that is scalable, compatible with missing values of data (robust), performs near real-time and is applicable in distributed platforms such as the cloud. These challenges are not adequately answered by traditional optimization methods, and the ultimate purpose of any modified or new optimization algorithm in the Big Data era is to reduce the computational, storage, and communications bottlenecks. One of the open issues faced by the data community is how to scale up analytic algorithms. To address this issue, we introduce a fully distributed stochastic optimization algorithm for decision making over large-scale data sets. We describe the proposed model mathematically. Our method is scalable to any network or data size, works based on the cooperation of neighbor processing/storage units, and it is adaptive to any dynamic behavior of processing/storage units. Experimental results on cognitive networks confirm that the proposed method performs well, proven to be distributed, scalable and robust to missing data and communication failures.

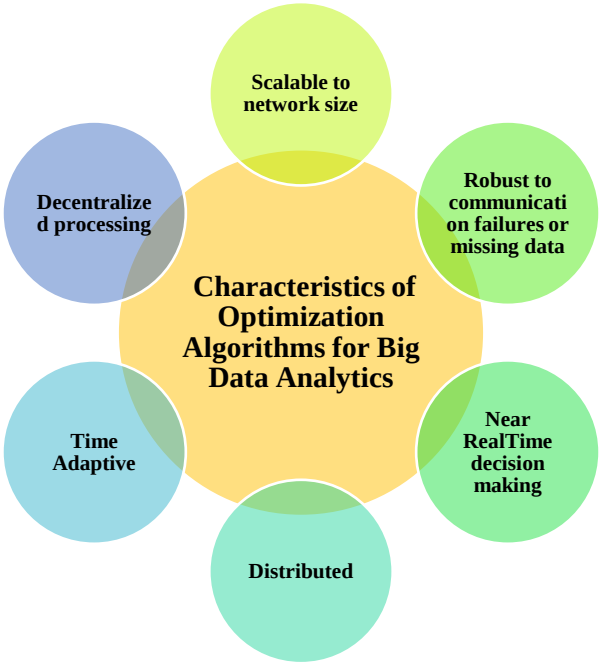

Figure 3-1 Different Characteristics of optimization algorithms for Big Data analytics

3.1.1 System Model

From the data point of view, there are two different approaches namely, centralized and distributed [38]. In centralized techniques, the data is transferred to a center for further processing/storage, whereas in a distributed manner, the data is exchanged and processed within the network locally. Transmitting the data to a center may cause network congestion and waste of communication and power resources. Any malfunction in the center causes network breakdown. Besides, the center requires high computation power to process the large volume of collected data. In comparison, in a distributed approach, the computational network load is divided between processing/storage units using cooperation and no centralized infrastructure is required.

The following notations are used throughout this chapter. Matrices are represented by upper case and vectors by lower case letters. Boldface fonts are reserved for random variables, and regular fonts are used for deterministic quantities. Superscript $(.)^T$ denotes transposition for real-valued vectors and matrices while $(.)^*$ denotes conjugate transposition for complex-valued vectors and matrices. The symbol $\mathbf{E}[.]$ is the expectation operator, $Tr(.)$ represents the trace of its matrix argument. I_M Represents the identity matrix of order M .

We consider a network consisting of N processing/storage units, also called " node" from now-on. The nodes are assumed to be distributed, each capable of processing and storing the limited size of data (at-least during the processing) and may or may not be involved in initial data generation. We assume that neighbor units can communicate with each other by using direct connection interfaces. Node l is said to be a neighbor of the node k if they can communicate and cooperate. We denote the set of all neighbors of the node k by N_k.

In this model, we assume that the nodes are generating or receiving continuous data with Big Data characteristics. It is impossible to transfer and process the data in a centralized manner because of the challenges faced by communication, security, time, and storage. The objective of the nodes in the network is to decide in a fully distributed manner. In other words, the solution is an estimate of an unknown parameter vector ω^o in a distributed way through stochastic optimization. At every time instant (iteration), i, each node k observes a scalar random process $\mathbf{d}_k(i)$ and a random vector process $\mathbf{u}_{k,i}$ which is related to ω^o via the linear regression model presented as follows [42]:

$$\mathbf{d}_k(i) = \mathbf{u}_{k,i}\omega^0 + \mathbf{v}_k(i) \tag{3-1}$$

The regression data $\{\mathbf{u}_{k,i}\}$ is zero mean, independent and identically distributed (i.i.d.) in time and independent over space with covariance matrices $R_{u,k} = \mathbf{E}\left[\mathbf{u}_{k,i}^{*}\mathbf{u}_{k,i}\right] > 0$. The noise $\mathbf{v}_k(i)$ is zero mean, i.i.d. in time and independent over space with variances $\sigma_{v,k}^2$. The $\mathbf{u}_{k,i}$ and the noise $\mathbf{v}_k(i)$ are mutually independent. The network will try to estimate ω^o by searching for the minimized global cost function, as presented below.

$$J^{glob}(\omega) = \sum_{k=1}^{N} \mathbf{E}\left|\mathbf{d}_k(i) - \mathbf{u}_{k,i}\omega\right|^2 \tag{3-2}$$

The most critical issue to solve an optimization problem in a distributed manner is to be able to separate the cost function among processing units. Each processing/storage unit should be able to act on its own while cooperating with neighbor nodes. Moreover, we assume that the cost function is separable among all processing units as follows.

$$J^{glob}(\omega) = \sum_{k=1}^{N} J_k(\omega) \tag{3-3}$$

$J_k(\omega)$ is the cost function of k processing/storage units defined as follows:

$$J_k(\omega) = \mathbf{E}\left|\mathbf{d}_k(i) - \mathbf{u}_{k,i}\omega\right|^2 \tag{3-4}$$

The cost function $J_k(\omega)$ can further be written in another form, as presented follows.

$$J_k(\omega) = \left\|\omega - \omega^o\right\|_{R_{u,k}}^2 + mmse_k \tag{3-5}$$

where $\|x\|_\Sigma^2$ denotes the weighted square quantity as $x^*\Sigma x$ for any semi-definite matrix $\Sigma \geq 0$ $R_{u,k} = \mathbf{E}\left[\mathbf{u}_{k,i}^* \mathbf{u}_{k,i}\right] > 0$ and $mmse_k$ is an additional MMSE term that is independent of ω.

Therefore, we have:

$$J^{global}(\omega) = J_k(\omega) + \sum_{l \neq k}\left(\left\|\omega - \omega^o\right\|_{R_{u,l}}^2 + mmse_l\right) \tag{3-6}$$

The optimum value ω^o that appears in the quadratic parts is not known. It should also be mentioned that the weighting matrices $R_{u,l}$ are not available in general, and only those from the neighbors can be assumed to be available. Therefore, we may conclude:

$$J_k^{dist}(\omega) = J_k(\omega) + \sum\nolimits_{l \in N_k \setminus \{k\}}\left(\left\|\omega - \omega^o\right\|_{R_{u,l}}^2\right) \tag{3-7}$$

Please note that the term $mmse_l$ is ignored since it is independent of ω and has no effects in finding the optimal value ω^o. The covariance matrices $R_{u,l}$ are not available in practice. Usually, processing/storage units can only observe realizations $\mathbf{u}_{l,i}$ of data arising from distributions whose covariance matrix is unknown $\left\{R_{\{u,l\}}\right\}$. One way to address this issue is to replace each of the weighted norms by a scaled multiple of the form as presented as follows:

$$\left\|\omega - \omega^o\right\|_{R_{u,l}}^2 \approx b_{l,k}\left\|\omega - \omega^o\right\|^2, \tag{3-8}$$

where $b_{l,k}$ is a non-negative coefficient? Each node k approximates the moment $R_{u,l}$ from its neighbors by multiples of the identity matrix. This Approximation is reasonable because using the Rayleigh-Ritz characterization of eigenvalues, it holds that:

$$\lambda_{\min}\left(R_{u,l}\right)\left\|\omega - \omega^o\right\|^2 \leq \left\|\omega - \omega^o\right\|_{R_{u,l}}^2 \leq \lambda_{\max}\left(R_{u,l}\right)\left\|\omega - \omega^o\right\|^2 \tag{3-9}$$

Therefore, we may conclude that:

$$J_k^{dist}(\omega) \approx J_k(\omega) + \sum\nolimits_{l \in N_k \setminus \{k\}} b_{l,k}\left\|\omega - \omega^o\right\|^2 \tag{3-10}$$

This new cost function at node k relies only on available information from neighbor nodes. Now, each k can apply a steepest-descent iteration to minimize the cost function as presented as follows:

$$\begin{aligned}\boldsymbol{\omega}_{k,i} &= \boldsymbol{\omega}_{k,i-1} - \mu_k \left[\nabla_{\omega} J_k^{dist}(\omega)\right]^* \\ \boldsymbol{\omega}_{k,i} &= \boldsymbol{\omega}_{k,i-1} + \mu_k \left(r_{du,k} - R_{u,k}\boldsymbol{\omega}_{k,i-1}\right) - \mu_k \sum_{l\in N_k\setminus\{k\}} b_{l,k}\left(\boldsymbol{\omega}_{k,i-1} - \omega^o\right)\end{aligned} \tag{3-11}$$

Where ∇_{ω} denotes the gradient vector. The step size parameters μ_k can be constant or variant. Fixed step size allows the algorithms to work continuously, while various step sizes that decay to zero causes the algorithms to stop after a while. Adaptive implementation of can be obtained by replacing covariance matrices by instantaneous approximations as presented:

$$\begin{aligned}r_{du,k} &\approx \mathbf{d}_k(i)\mathbf{u}_{k,i}^* \\ R_{u,k} &\approx \mathbf{u}_{k,i}^*\mathbf{u}_{k,i}\end{aligned} \tag{3-12}$$

Finally, by some substitution of equations, we may conclude that:

$$\boldsymbol{\omega}_{k,i} = \boldsymbol{\omega}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^*\left(\mathbf{d}_k(i) - \mathbf{u}_{k,i}\boldsymbol{\omega}_{k,i-1}\right) - \mu_k \sum_{l\in N_k\setminus\{k\}} b_{l,k}\left(\boldsymbol{\omega}_{k,i-1} - \omega^o\right) \tag{3-13}$$

The last correction term still depends on the unknown ω^o. Choosing different approximations ω^o leads to different strategies, such as consensus [42].

3.2 Proposed Method

Here, we first present the proposed method by its mathematical model and discuss the computational complexity of the presented method. In the proposed method, we apply diffusion adaptation [43] and by defining an intermediate variable, ψ, we have:

$$\begin{aligned}\psi_{k,i} &= \boldsymbol{\omega}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^*\left(\mathbf{d}_k(i) - \mathbf{u}_{k,i}\boldsymbol{\omega}_{k,i-1}\right) \\ \boldsymbol{\omega}_{k,i} &= \psi_{k,i} - \mu_k \sum_{l\in N_k\setminus\{k\}} b_{l,k}\left(\boldsymbol{\omega}_{k,i-1} - \omega^o\right)\end{aligned} \tag{3-14}$$

The unknown term ω^o is still shown in the equation. Considering $\psi_{l,i}$ as a substitute for ω^o we have:

$$\begin{aligned}\psi_{k,i} &= \boldsymbol{\omega}_{k,i-1} + \mu_k \mathbf{u}_{k,i}^*\left(\mathbf{d}_k(i) - \mathbf{u}_{k,i}\boldsymbol{\omega}_{k,i-1}\right) \\ \boldsymbol{\omega}_{k,i} &= \psi_{k,i} - \mu_k \sum_{l\in N_k\setminus\{k\}} b_{l,k}\left(\psi_{k,i} - \psi_{l,i}\right)\end{aligned} \tag{3-15}$$

It should be noted that in previous methods, $\boldsymbol{\omega}_{l,i-1}$ is usually substituted as ω^o. Defining $a_{l,k}$ as a weighting coefficient as presented and μ_k , we may conclude that:

$$a_{l,k}=\begin{cases}1-\sum_{j\in N_k\setminus\{k\}}\mu_k b_{j,k}, l=k\\ \mu_k b_{l,k,}, l\in N_k\setminus\{k\}\\ 0, otherwise\end{cases} \tag{3-16}$$

$$\sum_{i=1}^{\infty}\mu_k(i)=\infty, \sum_{i=1}^{\infty}\mu_k^2(i)<\infty \tag{3-17}$$

$$\psi_{k,i}=\boldsymbol{\omega}_{k,i-1}+\mu_k\mathbf{u}_{k,i}^*\left(\mathbf{d}_k(i)-\mathbf{u}_{k,i}\boldsymbol{\omega}_{k,i-1}\right)$$
$$\boldsymbol{\omega}_{k,i}=\sum_{l\in N_k}a_{l,k}\psi_{l,i-1}$$

The equation (3-17) could also be written in another form as presented as (3-19) where $c_{l,k}$ are the entries of the right-stochastic matrix $\mathbf{C}$, satisfying (3-18):

$$c_{l,k}\geq 0,\quad \mathbf{C1}_N=\mathbf{1}_N,\quad c_{l,k}=0\quad if\quad l\in N_k \tag{3-18}$$

$$\psi_{k,i}=\boldsymbol{\omega}_{k,i-1}+\mu_k\sum_{l\in N_k}c_{l,k}\mathbf{u}_{l,i}^*\left(\mathbf{d}_l(i)-\mathbf{u}_{l,i}\boldsymbol{\omega}_{l,i-1}\right) \tag{3-19}$$
$$\boldsymbol{\omega}_{k,i}=\sum_{l\in N_k}a_{l,k}\psi_{l,i-1}$$

3.2.1 Computational Complexity

Considering n, the average number of neighbor nodes in each iteration and assuming that total I iterations are needed for the convergence and having N processing/storage units in the network, the computational complexity is presented in Table 1. Comparing with convex optimization and considering semidefinite programming (SDP) or Second Order Cone Programming (SOCP) the complexity is non-linearly related to the number of data generator/processor/storage units, at-least by $o(N^2)$ [41].

Table 1. Computational Complexity of Proposed Distributed Method

Number of Additions	Number of Multiplications
(n+2)NI	2NI

3.3 Evaluation Results

In this part, we evaluate the proposed algorithm using a practical example. In this application, we define an optimization problem and assess the performance in terms of accuracy and robustness.

3.3.1 Cognitive Networks

One practical example of Big Data may be found in wireless sensor networks where the sensors generate a massive amount of data in a non-stop manner [44]. In wireless networks, it is shown that only a partial spectrum is used by the users [44]. Therefore, the cognitive systems are proposed as a solution and to improve the spectrum usage efficiency. Such systems include two types of users namely, primary and secondary. The primary users are the owner of the spectrum, and secondary users should continuously scene the spectrum (called spectrum sensing). When they find the unused band of spectrum for a while, use it based on the network predefined policy. Now, assume that the sensors should sense the data (level of energy available in the working frequency domain) and send their observations to a center. Besides the security, power consumption and data processing issues, the latency introduced in transferring data is not acceptable. This might be a straightforward application of the proposed method to make a decision continuously with high accuracy and reliability against any communication failure [44]. A simple diagram of a distributed spectrum sensing procedure is presented in Figure 3-2.

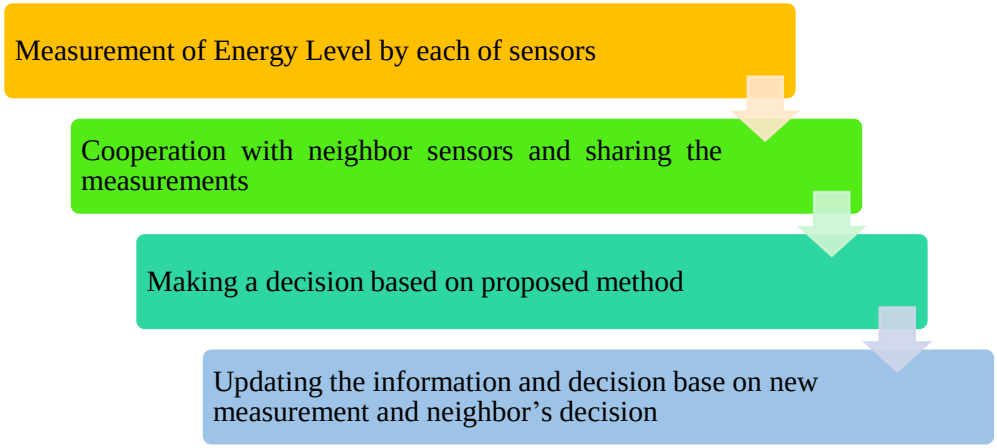

Figure 3-2 Instruction of performed tasks in distributed spectrum sensing

As presented in Figure 3-3, first, each sensor needs to measure the energy level and cooperate with other neighbors to make a distributed decision with local information. We consider a sample network consisting of 15 sensors (secondary users).

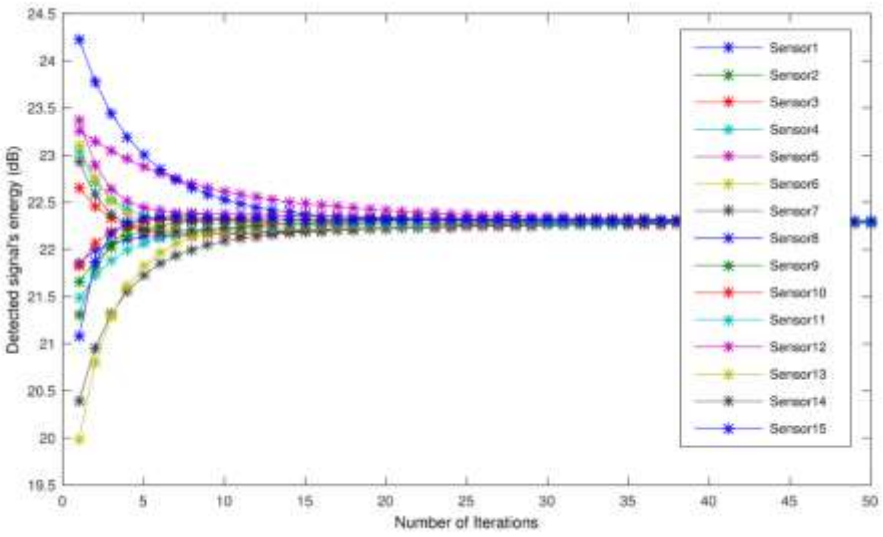

Figure 3-3 Performance of the distributed spectrum sensing when the communication link is ideal

We consider two scenarios; in the first scenario, we assume that the communication link between sensors (neighbors) is ideal. In this case, each sensor sends the data to its neighbors and makes a decision accordingly. The simulation results are presented in Figure 3-3. As illustrated in Figure 3-3, the first decision of each sensor is different. It is because each sensor has only access to its

local information. The challenges forced by communication, security, processing power and more critical time of the decision, make it impossible to gather the information in a center and process them simultaneously. We assume that information is transferred only to neighbors, and nodes perform the proposed optimization algorithm. After a few iterations, the whole network (each node) can reach a correct decision while each has only processed local information. This simulation shows how the continuously generated a large amount of data that is impossible to transfer and process in any of the nodes is processed in a fully distributed manner.

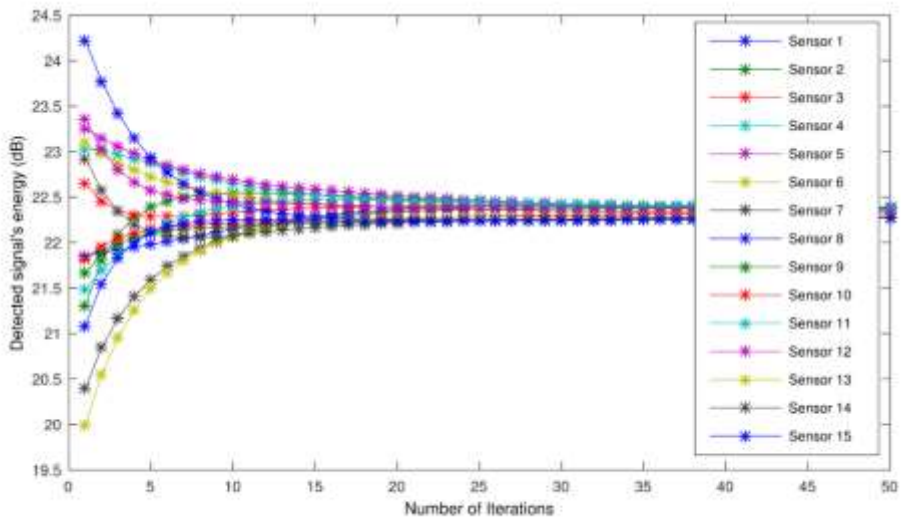

Figure 3-4 Performance of the distributed spectrum sensing when the communication link fails with the probability of 0.4

In the second scenario, we consider a more practical example. We assume that the communication between neighbor sensors is imperfect, meaning that we evaluate the proposed method when some data is missing. In this scenario, we try to assess the robustness of the proposed algorithm. We set the probability of communication failure to 0.4. It means that in each time instant (iteration), the likelihood of successful transmission is 0.6, and with a 40% chance, the transmitted data is missing. The result is presented in Figure 3-4.

As simulation results indicate, the proposed stochastic optimization method finds the global optimum while only local information is exchanged through the network. The convergence of this network means that, although sensors only process their cost function, the information diffuses to the network. Considering the communication link imperfection, the method is robust against missing data. It should be mentioned that the algorithm is capable of being used in all networks with arbitrary size. The simulation is performed with Matlab and part of the simulation code is presented in the Appendix.

3.3.2 Conclusion

In this chapter, we presented a fully distributed method to decide large-scale networks and data sets. The proposed method is scalable to any network configuration, is near real-time (in each iteration, a solution is achieved although this solution is not precise) and more critical, robust to any missing data or communication failures. We evaluated the proposed method by a practical example and simulations on cognitive networks. Simulation results confirmed that the proposed method is efficient in terms of accuracy and robustness. Here, we evaluated the proposed algorithm with a simple application of cognitive sensor networks.

4 Chapter 4

Optimal Weight Design Algorithm for decision making over large-scale data sources

4.1 Introduction

The nature of decentralized large-scale data-sources requires distributed algorithms. In previous chapter, we proposed an algorithm in which it is assumed that the data-sources are capable of processing their data and collaborating with neighbor sources. We assumed that the network objective is to make an optimal decision while the data is processed in a distributed manner. New technologies, such as the next generation of wireless communication and 5G [7, 52], introduce practical issues, including imperfect communication that needs to be addressed in the context of Big Data.

In this chapter, we propose an optimal algorithm that uses optimal weighting to combine the resource of neighbors. We model the effect of imperfect communication on an optimization problem and find the solution by applying the proposed algorithm. We evaluate the performance of the developed algorithm by using both mathematical methods and computer simulations. We introduce the conditions in which the convergence of the proposed algorithm is guaranteed and prove that the network error decreases considerably compared with some of the known modern methods.

We assume that each source is capable of communicating and processing a limited amount of data. An enormous amount of sensing devices collect or generate sensory data over time for a wide range of applications witch results in Big Data [51]. One example of such networks could be found in smart-city or Internet of things (IoT) concept in-which sensory data is transmitted to Edges or clouds. The network needs to make a decision in near real-time and broadcast the decision to network actuators as depicted in Figure 4-1.

It is not practical to use centralized solutions because of Big Data characteristics in large-scale networks. The recently distributed algorithms, usually focus on the performance, neglecting the practical issues. For instance, trying to decide on the minimum time, using load-balancing algorithms is mandatory; because of the dynamic characteristics of data-sources in volume, processing capabilities, and so forth. One of the critical features in distributed algorithms is collaboration; meaning that the data-sources are capable of sharing their initial decision and part of their data with some of the neighbor sources. This property is the key to reach a global decision over the network, while the data is processed in a fully distributed manner. Until now, the communication link between the data-sources is considered ideal, meaning that the effect of communication link on the data transfer is neglected. However, new wireless communication technologies such as 5G and the next generations of wireless communication are considered as

promising solutions to form the communication infrastructure of large-scale networks. These wireless links introduce noise, path-loss, multi-path and fading that damage the data.

Here, we focus on fully distributed data-analytics algorithms and add a practical constraint to the optimization problem. We assume that the communication between data-sources is non-ideal. We model the effect of this non-ideal channel on data and consider an optimization problem and find the solution by using a distributed decision-making algorithm with optimal weighting.

Figure 4-1 Sample diagram of the sensory network in-which the data is transferred to cloud/Edge data-sources that can communicate. The network should make a decision or insight from such Big Data in a distributed manner.

The following properties are assumed for data sources:

- Data-sources rely only on their local information and observations.
- Data-sources can exchange some limited information to near neighbors.
- The network is dynamic, and the algorithms should be robust to the changes in the network topology.
- Sources are capable of computing limited information

Considering the mentioned challenges, researchers are trying to find optimized distributed decision-making algorithms. In such algorithms, it is assumed that the decentralized data from a variety of sources could be utilized to make an optimized decision for the network while all the contributors benefit. The main idea of this solution lies behind several assumptions, listed in the following.

• All data-sources can make an initial decision by themselves (can process a limited amount of data).

• All data sources are capable of sharing their own decisions and part of their data (for example, in a vehicular network, the sensory data) with some neighbors in the network.

• The decision-making objective is accepted among all the data-sources. Each source is capable of making its own decision only by using its gathered data while contributes to the network's global decision.

Finding the optimal solution, which guarantees the best convergence rate, convergence area, and minimum cost, is still an open issue. In this section, we try to improve the performance of the existing algorithms by proposing optimal weight design over the Internet of things applications that regularly face some practical challenges, including imperfect communication channels.

4.1.1 System Model

In this section, we present the system model At first we define the variable sets and notation followed by the mathematical representation of distributed decision-making model.

4.1.1.1 Notation

The following notations are used throughout this paper. Matrices are represented by upper case and vectors by lower case letters. Boldface fonts are reserved for random variables, and regular fonts are used for deterministic quantities. Superscript $(.)^T$ denotes transposition for real-valued vectors and matrices while $(.)^*$ Denotes conjugate transposition for complex-valued vectors and matrices. The symbol $E[.]$ is the expectation operator, $Tr(.)$ represents the trace of its matrix argument. I_M Represents the identity matrix of order M . The $diag\{.\}$ operator shows a diagonal matrix and $vec(.)$ the operator creates a vector from a matrix. $bvec(.)$ the operator first takes the

blocks of a matrix and make a vector from this block by serializing the data. $\otimes$ shows the Kronecker product and $\otimes_b$ represents the block-Kronecker product [53].

We consider a network consisting of N decentralized data-sources, each capable of processing a limited amount of data and communicate with its neighbors. The set of all neighbors of a source k is denoted by N_k. The sources are generating or accumulating data with some of Big Data properties. The objective of the network is to decide in a fully distributed manner. Mathematically speaking, the solution is to find an unknown vector ω^o, as a distributed estimation, that is performed by using iterative techniques. In such techniques, at each time instant (iteration) i, sources have access to the following two types of data: a scalar random process $d_k(i)$ and a vector random process $u_{k,i}$ both related to ω^o where s $v_k(i)$ is data gathering noise [54].

$$d_k(i) = u_{(k,i)}\omega^0 + v_k(i) \tag{4-1}$$

In equation (4-1) it is assumed that the:

• $\{u_{k,i}\}$ is zero-mean, independent, and identically distributed (i.i.d.) in time and independent over space.

• the covariance matrices are shown with $R_{u,k} = E\left[u_{k,i}^* u_{k,i}\right] > 0$.

• The noise $v_k(i)$ is zero-mean, i.i.d. in time and independent over space with variances $\sigma_{v,k}^2$

• The $u_{k,i}$ and the noise $v_k(i)$ are mutually independent.

As an example, consider an IoT network consisting of many devices that transfer their sensory data to near edge or clouds as depicted in Figure 4-1. Therefore, the data-sources have access to instant observations divided as scalar or vector random processes. The objective could be a decision on choosing the optimal power flow or path suggestion. The data is produced with high volume, velocity, and variety, and it is not possible to make a network decision (global decision) in near real-time.

4.1.2 Optimal-Weight Distributed Algorithms

In general, we try to solve the equation (4-2) in an optimized manner. To find the optimal weighting matrix, we assume that the $a_{l,k}(i) = \gamma_{l,k} I_{l,k}(i)$ where $\gamma_{l,k}$ are positive fixed combination

weights that data-source k assigns to l^{th} Neighbor. The $I_{l,k}(i)$. Pretenses the network communication topology and is equal to 1 when the two data-sources collaborate (communicate) and 0 otherwise. Some of the known optimal weighting algorithms in the literature are Metropolis [55], as presented in equation (4-3) and Laplacian adaptive methods [56] as given in equation (4-4).

$$\omega_{(k,i)} = \sum_{l\in N_k} a_{l,k}\omega_{l,i} + \mu_k u^{*}_{k,i}\left(d_k(i) - u_{k,i}\omega_{k,i-1}\right) \tag{4-2}$$

$$a_{l,k}(i) = \begin{cases} \dfrac{1}{\max\left\{\left|N_{k,i}\right|,\left|N_{l,i}\right|\right\}} & if \quad l \in N_{k,i}\setminus\{k\} \\ 1-\sum_{l\in N_{k,i}\setminus\{k\}} a_{l,k}(i), & if \quad l = k \\ 0, \quad otherwise \end{cases} \tag{4-3}$$

$$a_{l,k}(i) = \begin{cases} \dfrac{1}{\max\left\{\left|N_{k,i}\right|, k \in \{1,2,...,N\}\right\}} & if \quad l \in N_{k,i}\setminus\{k\} \\ 1-\sum_{l\in N_{k,i}\setminus\{k\}} a_{l,k}(i), & if \quad l = k \\ 0, \quad otherwise \end{cases} \tag{4-4}$$

Although these methods are modified to take into account different network topologies, they still are not the optimal solution for the networks with imperfect communication among sources. In what follows, we first present the effect of imperfect communication in damaging the data and then describe the propose optimal weighting algorithm.

4.1.3 Effect of Imperfect Communication

We assume that the $\psi_{lk,i}$ is the estimate of received data on data-source k, sent from the source l over the imperfect wireless channel (for instance using 5G [52]) in which the communication channel affects the data by $h_{l,k}$ as follow:

$$\psi_{lk,i} = h_{l,k}(i)\sqrt{\frac{P_t}{r_{l,k}^{\alpha}}}\psi_{l,k} + v_{lk,i}^{(\psi)} \tag{4-5}$$

where P_t is the power of the transmitted signal and $r_{l,k} = r_{k,l}$ is the distance between the source l and k. α is the path-loss component and $v_{lk,i}^{(\psi)}$ is the additive zero-mean noise vector with $\sigma_{v_{lk,i}}^{(\psi)2} I_M$ as its covariance matrix. The channel coefficients are space-independent and time-variant circular Gaussian random variables with zero-mean and $\sigma_{h_{l,k}}^2 = 1$. The noise vector, $\left\{v_{lk,i}^{(\psi)}\right\}$ is also zero-mean and i.i.d in time. The channeling effect ($h_{l,k}$) could be neglected only when the signal is stronger than a pre-defined threshold, $\varsigma_{lk}^0 \Delta = \frac{P_t}{\sigma_{v_{lk,i}}^{(\psi)2} r_o^{\alpha}}$.

Considering the channel effect, the $\varsigma_{lk}(i) = \frac{\left|h_{lk}(i)\right|^2 P_t}{\sigma_{v_{lk,i}}^{(\psi)2} r_{l,k}^{\alpha}}$ Is defined. The data transmission is defined as successful when $\varsigma_{lk}(i) \geq \varsigma_{lk}^o$ meaning that the $\left|h_{lk}(i)\right|^2 \geq v_{l,k}$.

While the channel coefficients, $h_{lk}(i)$ are circular Gaussian random variables, the $\left|h_{lk}(i)\right|^2$ follows exponential distribution with $\lambda = 1$. Therefore, the probability of successful transmission could be modeled as follows:

$$p_{l,k} = P_r\left(\left|h_{lk}(i)\right|^2 \geq v_{l,k}\right) = e^{-v_{l,k}} \tag{4-6}$$

This demonstrates that increasing the distance between data-sources while the transmission power is constant, the chance of successful data transmission decreases.

We assume that network topology is dynamic, meaning that the neighbors may vary during the iterations. Therefore, $N_{k,i} \subset N_k$ is defined as a subset of neighbors to data-source k at the iteration index i. Therefore, the main calculation formula in equation (4-2) could be re-written as follows:

$$\omega_{k,i} = \omega_{k,i-1} + \mu_k u_{k,i}^* \left(d_k(i) - u_{k,i}\omega_{k,i-1}\right) - \mu_k \sum_{l \in N_{k,i} \setminus \{k\}} b_{l,k}(i)\left(\omega_{k,i-1} - \omega^o\right) \tag{4-7}$$

4.2 Proposed Optimal Weight Design

The weights, in general, could be presented as equation (4-8).

$$a_{l,k}(i)=\begin{cases}\gamma_{l,k} I_{l,k}(i), & if \quad l \in N_{k,i} \setminus \{k\} \\ 1-\sum_{l \in N_{k,i} \setminus \{k\}} a_{l,k}(i), & if \quad l=k \\ 0, & otherwise\end{cases} \tag{4-8}$$

where $\sum_{l \in N} \gamma_{l,k}=1$, $\sum_{l \in N-k_i} \gamma_{l,k}<1$. To find the optimal weight, we need to find the combination that minimizes the network error. As long as the A_i satisfies $\sum_{l \in N_{k,i} \setminus \{k\}} a_{l,k}(i)=1$ we have:

$$\mathbf{E}[A_i]=\mathbf{E}\left[\sum_{l \in N_{k,i}} a_{l,k}(i) I_{l,k}(i)\right]=\lim_{i \to \infty} \frac{1}{i} \sum_{j=1}^{i} \sum_{l \in N_{k,i}} a_{l,k}(i) I_{l,k}(i)=1 \tag{4-9}$$

It is straightforward to prove that $E\tilde{\omega}_{i\sigma}^2=E\left[\tilde{\omega}_{F^{i+1}\sigma}^2\right]+\gamma^T \sum_{j=0}^{i} F^j \sigma$ where the $\tilde{\omega}$ Is the error vector.

Proof:

Defining $\tilde{\boldsymbol{\omega}}_i=B_i \tilde{\boldsymbol{\omega}}_{i-1}-My_i$ we have:

$$\mathbf{E}\left[\left\|\tilde{\boldsymbol{\omega}}_i\right\|_{\sum}^2\right]=\mathbf{E}\left[\left\|\tilde{\boldsymbol{\omega}}_{i-1}\right\|_{\sum'}^2\right]+\mathbf{E}\left[\mathbf{y}_i^* \sum \mathbf{y}_i\right] \tag{4-10}$$

Where $\sum \geq 0$ is the arbitrary weighting matrix and $\sum' = B_i^* \sum B_i$. Since, $\tilde{\boldsymbol{\omega}}_{i-1}$ and R_i are statistically independent, we have:

$$\mathbf{E}\left[\left\|\tilde{\boldsymbol{\omega}}_{i-1}\right\|_{\sum'}^2\right]=\mathbf{E}\left[\left\|\tilde{\boldsymbol{\omega}}_{i-1}\right\|_{\mathbf{E}\left[\sum'\right]}^2\right] \tag{4-11}$$

$$\mathbf{E}\left[\left\|\tilde{\boldsymbol{\omega}}_i\right\|_{\sum}^2\right]=\mathbf{E}\left[\left\|\tilde{\boldsymbol{\omega}}_{i-1}\right\|_{\sum'}^2\right]+Tr\left[\sum Y\right] \tag{4-12}$$

where:

$$\begin{aligned}
&\sum{}' = \mathbf{E}\left[B_i^* \sum B_i \right], \\
&Y \triangleq \mathbf{E}\left[y_i y_i^* \right], \\
&Y = A^T MGMA, \\
&G = diag\left\{ \sigma_{v,1}^2 R_{u,1}, ..., \sigma_{v,N}^2 R_{u,N} \right\}.
\end{aligned} \tag{4-13}$$

Assuming X, U and V as $NM \times NM$ matrices, we have:

$$\begin{aligned}
&vec\left(U \sum V\right) = \left(V^T \otimes U\right) vec\left(\sum\right) \\
&vec\left(\sum X\right) = \left| vec\left(X^T\right) \right| vec\left(\sum\right)
\end{aligned} \tag{4-14}$$

Defining $\sigma = vec\left(\sum\right)$ and $\sigma' = vec\left(\sum{}'\right)$ we have $\sigma' = F\sigma$ where F is definable. Therefore, we have:

$$\mathbf{E}\left\| \tilde{\boldsymbol{\omega}}_i \right\|_\sigma^2 = \mathbf{E}\left[\left\| \tilde{\boldsymbol{\omega}}_{i-1} \right\|_{F\sigma}^2 \right] + \gamma^T \sigma \tag{4-15}$$

where $\sum$ is substituted by σ, $\gamma = vec\left(Y^T\right)$ and $F = \mathbf{E}\left| B_i^T \otimes B_i^* \right|$.

Finally, it is proven that $\mathbf{E}\left\| \tilde{\boldsymbol{\omega}}_i \right\|_\sigma^2 = \mathbf{E}\left[\left\| \tilde{\boldsymbol{\omega}}_{-1} \right\|_{F^{i+1}\sigma}^2 \right] + \gamma^T \sum_{j=0}^{i} F^j \sigma$.

The optimization problem is to find the weighting matrix that minimized the network error. Since we do not have access to the exact error; we try to minimize the upper-bound of the network error [49,50]. Therefore, the value η is defined as equation (4-16):

$$\begin{aligned}
&\eta = \lim_{i \to \infty} \frac{1}{N} \mathbf{E}\left\| \tilde{\omega}_i \right\|_{bvec(I_{MN})}^2 \\
&\eta = \lim_{i \to \infty} \frac{1}{N} \left\| \tilde{\boldsymbol{\omega}}_{-1} \right\|_{F^{i+1} bvec(I_{MN})}^2 + \lim_{i \to \infty} \frac{1}{N} \sum_{j=0}^{i} Tr\left(B^j \left(\mathbf{E}\left[A_i^T MPMA_i \right] + R_u \right) B^{*j} \right)
\end{aligned} \tag{4-16}$$

where:

$$\begin{aligned}
&B \triangleq \mathbf{E}\left[B_i \right] \\
&F \approx B^T \otimes_b B^* \\
&\tilde{\boldsymbol{\omega}}_{k,i} = \omega^o - \boldsymbol{\omega}_{k,i} \\
&G_i \triangleq col\left\{ \mathbf{u}_{1,i}^* \mathbf{v}_1(i), ..., \mathbf{u}_{N,i}^* \mathbf{v}_N(i) \right\} \\
&R_i \triangleq diag\left\{ \mathbf{u}_{1,i}^* \mathbf{u}_{1,i}, ..., \mathbf{u}_{N,i}^* \mathbf{u}_{N,i} \right\} \\
&M \triangleq diag\left\{ \mu_1 I_M, ..., \mu_N I_M \right\} \\
&A = A \otimes I_M
\end{aligned} \tag{4-17}$$

It is also proven that when B is steady, $\lim_{i\to\infty}\frac{1}{N}\left\|\tilde{\boldsymbol{\omega}}_{-1}\right\|^2_{F^{i+1}bvec(I_{MN})}=0$ [49]. Therefore, we have:

$$\eta=\lim_{i\to\infty}\frac{1}{N}\sum_{j=0}^{i}Tr\left(B^j\left(\mathbf{E}\left[A_i^T MPMA_i\right]+R_u\right)B^{*j}\right) \tag{4-18}$$

Defining $X_*=\sum_j\sigma_j\left(X\right)$ as the nuclear norm of X we have:

$$\begin{aligned}
&Tr\left(\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right)\overset{i}{=}\left\|\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right\|_* \quad (1)\\
&Tr\left(\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right)\overset{ii}{\le}\left\|\mathcal{B}^j\right\|_*\left\|\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\right\|_*\left\|\mathcal{B}^{*j}\right\|_* \quad (2)\\
&Tr\left(\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right)\overset{iii}{\le}\left\|\mathcal{B}\right\|_*^{2j}Tr\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right) \quad (3)\\
&Tr\left(\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right)\overset{iv}{=}c^2\left\|\mathcal{B}\right\|_{b,\infty}^{2j}Tr\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right) \quad (4)\\
&Tr\left(\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right)\overset{v}{\le}c^2\left\|I-\mathcal{MR}\right\|_{b,\infty}^{2j}Tr\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right) \quad (5)\\
&Tr\left(\mathcal{B}^j\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right)\mathcal{B}^{*j}\right)\overset{vi}{=}c^2\left[\rho\left(I-\mathcal{MR}\right)\right]^{2j}Tr\left(\mathbb{E}\left[\mathcal{A}_i^T\mathcal{MPMA}_i\right]+\mathcal{R}_u\right) \quad (6)
\end{aligned} \tag{4-19}$$

Substituting the values, we have:

$$\eta\le\lim_{i\to\infty}\frac{c^2}{N}\sum_{j=0}^{i}Tr\left(E\left[A_i^T MPMA_i\right]+R_v\right)\sum_{j=0}^{i}\left[\rho\left(I-MR\right)\right]^{2j} \tag{4-20}$$

Considering equation (4-20), the optimization problem is defined as follows:

$$\begin{aligned}
&\min_A Tr\left(E\left[A_i^T MPMA_i\right]+R_v\right)\\
&s.t.\quad A1=1,\quad a_{l,k}\ge 0\\
&a_{l,k}=0\quad if\quad l\notin N_k
\end{aligned} \tag{4-21}$$

It is easily shown that:

$$\begin{aligned}
&Tr\left(E\left[A_i^T MPMA_i\right]+R_v\right)=\\
&\sum_{k=1}^{N}\sum_{l\in N_k}E\left[a_{l,k}^2(i)\right]\left\{\mu_l^2\sigma_{v,l}^2Tr\left(R_{u,l}\right)+E\left[\left|g_{l,k}(i)\right|^2\left|\left|h_{l,k}(i)\right|^2\right\rangle v_{l,k}\right]Tr\left(R_{v,lk}^{(\psi)}\right)\right\}
\end{aligned} \tag{4-22}$$

We also know that the columns of A_i are independent and A is LF, therefore:

$$Tr\left(E\left[A_i^T MPMA_i\right]+R_v\right)= \\ \sum_{k=1}^{N}\sum_{l\in N_k}E\left[a_{l,k}^2(i)\right]\left\{\mu_l^2\sigma_{v,l}^2 Tr\left(R_{u,l}\right)+E\left[\left|g_{l,k}(i)\right|^2\left|h_{l,k}(i)\right|^2\right\rangle v_{l,k}\right]Tr\left(R_{v,lk}^{(\psi)}\right)\right\} \quad (4\text{-}23)$$

$$E\left[a_{l,k}^2(i)\right]=\lambda_{l,k}^2 E\left[I_{l,k}^2(i)\right]=\gamma_{l,k}^2 p_{l,k} \quad (4\text{-}24)$$

The optimization problem could now be simplified as the equation (4-25).

$$\min_{\gamma_{l,k}} \sum_{l\in N_k}\gamma_{l,k}^2 p_{l,k}\left\{\mu_l^2\sigma_{v,l}^2 Tr\left(R_{u,l}\right)+\left|g_{l,k}\right|^2 Tr\left(R_{v,lk}^{(\psi)}\right)\right\} \\ s.t.\quad \gamma_{l,k}\geq 0,\quad \sum_{l\in N_k}p_{l,k}\gamma_{l,k}=1,\quad \gamma_{l,k}=0\quad if\quad l\notin N_k \quad (4\text{-}25)$$

where:

$$\left|g_{l,k}\right|^2=E\left[\left|g_{l,k}(i)\right|^2\left|h_{l,k}(i)\right|^2>v_{l,k}\right] \quad (4\text{-}26)$$

Applying KKT conditions and Lagrangian we have:

$$p_{l,k}\gamma_{l,k}=\begin{cases}\dfrac{p_{l,k}\alpha_{l,k}^{-2}}{\sum_{m\in N_k}p_{m,k}\alpha_{m,k}^{-2}}, & if\quad l\in N_k \\ 0, & otherwise\end{cases} \quad (4\text{-}27)$$

$E\left[I_{l,k}(i)\right]=p_{l,k}$ and finally, the optimal weights are presented in equation (4-28).

$$a_{l,k}(i)=\begin{cases}\dfrac{\alpha_{l,k}^{-2}}{\sum_{m\in N_{k,i}}\alpha_{m,k}^{-2}}, if l\in N_{k,i} \\ 0, otherwise\end{cases} \quad (4\text{-}28)$$

where $\alpha_{l,k}^2$ is defined as equation (4-29):

$$\alpha_{l,k}^2=\begin{cases}\mu_l^2\sigma_{v,l}^2 Tr\left(R_{u,l}\right)+M\left|g_{l,k}\right|^2\sigma_{v,lk}^{2(\psi)}, & if\quad l\in N_k\setminus\{k\} \\ \mu_k^2\sigma_{v,k}^2 Tr\left(R_{u,k}\right), & if\quad l=k\end{cases} \quad (4\text{-}29)$$

However, we have assumed that the $\sigma_{v,l}^2$ and $R_{u,l}$, $|g_{l,k}|^2$ and $\sigma_{v,lk}^{2(\psi)}$ are pre-known which is not practical, and the instant estimates are used instead.

4.2.1 Convergence Analysis

In this section, we analyze the convergence of the proposed algorithm. At first, some parameters need to be defined. $\tilde{\omega}_i$ is defined as network error vector and we have $\tilde{\omega}_i = A^T(I - MR_i)\tilde{\omega}_{i-1} - A^T MG_i$ where G_i R_i and M are intermediate matrices defined as follows:

$$\begin{aligned}
&\tilde{\omega}_{k,i} = \omega^o - \omega_{k,i} \\
&G_i\Delta = col\left\{u_{1,i}^* v_1(i), ..., u_{N,i}^* v_N(i)\right\} \\
&R_i\Delta = diag\left\{u_{1,i}^* u_{1,i}, ..., u_{N,i}^* u_{N,i}\right\} \\
&M\Delta = diag\left\{\mu_1 I_M, ..., \mu_N I_M\right\} \\
&A = A \otimes I_M
\end{aligned} \tag{4-30}$$

By defining the $B_i = A^T(I - MR_i)$ and $y_i = A^T MG_i$ the error vector may be presented as $\tilde{\omega}_i = B_i\tilde{\omega}_{i-1} - My_i$ in which B_i controls the dynamic error vector. To prove the convergence we need to prove that the $E[\tilde{\omega}_i] = BE[\omega_{i-1}]$ where $B\Delta = E[B_i] = A^T(I - MR)$ [57].

Considering $E[\tilde{\omega}_i] = BE[\omega_{i-1}]$, if $\lim_{i\to\infty} E[\tilde{\omega}_i] \to 0$ the convergence is proven. This requires that the B is steady. To have steady B, its spectral radius needs to satisfy $\rho(B) < 1$.

In general, it is not possible to prove $\rho(B) < 1$, however, the $u_{k,i}$ and $v_{k,i}$ values are independent and $v_{k,i}$ is zero-mean, therefore, $E\left[A^T MG_i\right] = 0$. We also know that A is an LF matrix and therefore, its spectral efficiency, $\rho(A) = 1$. A is also symmetric and therefore, all its elements are real values and therefore, $\lambda_{\max}(A) = \rho(A) = 1$.

Therefore, if we can find the conditions in which $\rho(B) < 1$ the convergence is guaranteed. On the other hand, considering the above-mentioned conditions on LF, $\rho(B) < 1$ holds that the $\rho(B) < \rho(I - MR)$. In [58], it is proven that if the step-sizes, μ_k chosen so that the $0 < \mu_k < \dfrac{2}{\lambda_{\max}(R_{u,k})}$ *for* $k = 1, 2, ..., N$ is satisfied, the $\rho(B) < \rho(I - MR)$ is guaranteed. Therefore,

choosing the step-sizes according to $0 < \mu_k < \frac{2}{\lambda_{\max}\left(R_{u,k}\right)}$ *for* $k = 1, 2, ..., N$ and considering the LF matrix of network, the convergence is guaranteed.

4.2.2 Evaluation Results

In this section, we evaluate the performance of the proposed optimal distributed decision-making algorithm, and we compare the weighting strategy with Metropolis and Laplacian combination methods. For simulations we use Matlab and part of the simulation codes are presented in Appendix.

We consider a network with ten decentralized data-sources, $N=10$ as depicted in Figure 4-2 The diagram of the network used for evaluation with 10 decentralized data-sources. We set maximum step-size to 0.005, the number of iterations to 50 and we average the results over 10 realizations. In the same network topology, we compare the proposed optimal weighting, Metropolis method [55] and also optimal weighting proposed in [56]. Figure 4-3 Normalized network error versus the number of iterations for the proposed method, Metropolis [55] and optimal combination method [56] presents the normalized network error versus the number of iterations for the proposed method, Metropolis [55] and optimal combination method [56]. As can be seen in figure 4-3 the performance of the proposed method shows considerable improvements. It is also shown that the optimal weighting does not reach the theoretical error bound in a limited number of iterations.

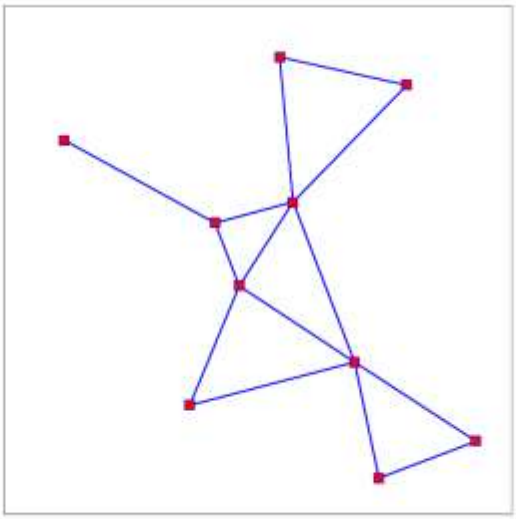

Figure 4-2 The diagram of the network used for evaluation with 10 decentralized data-sources

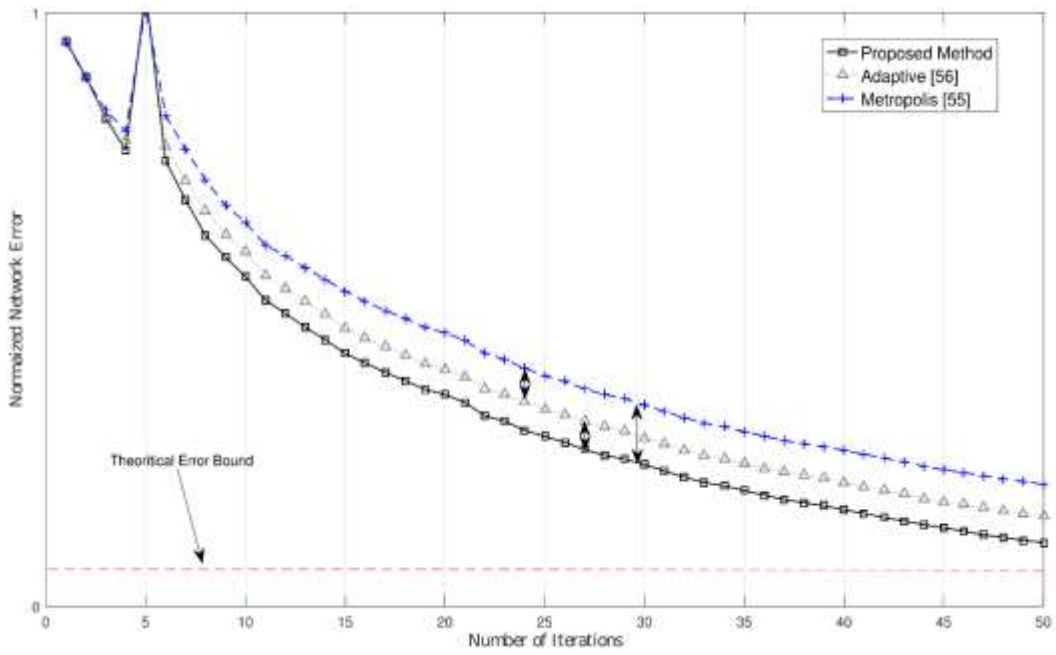

Figure 4-3 Normalized network error versus the number of iterations for the proposed method, Metropolis [55] and optimal combination method [56]

4.3 Conclusion

The nature of decentralized large-scale data-sources requires distributed algorithms. Distributed algorithms are more robust and secure in compared with centralized solutions, while they introduce new challenges such as communication failures. Distributed data-sources should be capable of processing their data and communicate with neighbor sources to find the network objective as an optimal decision. This process needs to be made in a distributed manner, with no need to implement a centralized system center and to have access only to local information. Some challenges are introduced by new technologies such as 5G or high-speed wireless data transfer, including imperfect communications that damage the data. In this chapter, we proposed an optimal algorithm, that uses optimal weighting to combine the shared data coming from neighbors. This optimal weight improves the performance of the decision-making algorithm in terms of error and convergence rate. We defined an optimization problem and proposed an optimization weighting algorithm to find a solution to the optimization problem. We evaluated the performance of the proposed algorithm mathematically and introduced the step-sized conditions that guaranteed the

convergence of the proposed algorithm. We also used computer simulations to assess the network error. We proved that in a network diagram with 10 data-sources, the network performance of the proposed algorithm outperforms some of the known optimal solutions such as Metropolis and adaptive combination.

5 Chapter 5

Conclusion and Future works

5.1 Summary and Conclusion

In chapter 1, we introduced the basics of what is going to be presented in this thesis, including the introduction to optimization, the definition of Big Data, its importance and history, and also the general relation of Big Data and optimization techniques.

In chapter 2 of this thesis, we reviewed some of the recent and relevant optimization algorithms used for Big Data analysis, including machine learning algorithms, classic optimization algorithms, and heuristic and evolutionary optimization algorithms. We also reviewed some of the most useful tools in Big Data analysis.

In chapter 3, we showed that distributed algorithms are a promising solution to the decentralized nature of Big Data algorithms. We presented the basics of stochastic distributed decision-making algorithms and proposed a fully distributed one that was based on diffusion adaptation. The proposed algorithm used the collaboration of neighbor nodes to make a global decision while the sources make local decisions. We evaluated the proposed algorithm on cognitive radio networks in which secondary users use energy detector sensors to sense the environment. We showed that the proposed algorithms perform well, is scalable and robust to communication failures.

In chapter 4, we considered some practical issues including the effect of imperfect communications on distributed analysis algorithms. We modeled this effect as an optimization problem and solved the problem to gain the optimal weights that each node assigns to the information received from neighbor nodes. We compared the proposed optimal weighing algorithm with some of the earlier methods. Evaluation results confirmed that the proposed method performs well in terms of error and convergence rate. We also presented mathematical analysis on the convergence of the proposed algorithm.

5.2 Future Works

In this thesis, we only considered practical issues in distributed optimization algorithms that are imperfect communication links. We assumed the cost function of the whole network (the decision that the whole network is going to make) is separable among all nodes (data-sources). These assumptions are usual in the literature. However, if we need to decide Big Data in almost near real-time, we need to consider some other issues including the computational capability of each source and data transfer delays. For instance, assume that some sources have stronger computational

resources. The total decision time depends on the weakest source. Therefore, local balancing techniques are required to optimize the algorithms in practical scenarios.

References

[1] Aragón, Francisco J., Miguel A. Goberna, Marco A. López, and Margarita ML Rodríguez. Nonlinear optimization. Springer International Publishing, 2019.

[2] Furht, Borko, and Flavio Villanustre. Big data technologies and applications. Berlin, Germany: Springer, 2016.

[3] A. L'heureux, K. Grolinger, H.F. Elyamany, M.A. Capretz: Machine learning with Big Data: Challenges and Approaches, IEEE Access, 5(5) 2017, 777-97.

[4] Stefanos Vrochidis, Benoit Huet, Edward Y Chang, and Yiannis Kompatsiaris. Big Data Analytics for Large-scale Multimedia Search. Wiley Online Library, 2019.

[5] K. Wang, Y. Wang, X. Hu, Y. Sun, D.J. Deng, A. Vinel, Y. Zhang: Wireless Big Data computing in smart grid, IEEE Wireless Communications, 24(2) 2017, 58-64.

[6] Zhu, Li, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. "Big data analytics in intelligent transportation systems: A survey." IEEE Transactions on Intelligent Transportation Systems 20, no. 1 (2018): 383-398.

[7] A.M. Somarin, M. Barari, H. Zarrabi: Big Data Based Self-Optimization Networking in Next Generation Mobile Networks, Wireless Personal Communications, 101(3) 2018, 1499-1518.

[8] Z. Gui-Xia, Z. Cheng-Jing, W. Xiao-Yan: Research of Distributed Data Optimization Storage and Statistical Method in the Environment of Big Data, In International Conference on Smart Grid and Electrical Automation (ICSGEA), IEEE, May 2017, 612-617.

[9] Barnes, Trevor J. "Big data, little history." Dialogues in Human Geography 3, no. 3 (2013): 297-302.

[10] Power, Daniel J. "Using 'Big Data'for analytics and decision support." Journal of Decision Systems 23, no. 2 (2014): 222-228.

[11] Plunkett, Tom, Brian Macdonald, Bruce Nelson, Mark Hornick, Helen Sun, Khader Mohiuddin, Debra Harding et al. Oracle big data handbook. McGraw-Hill Osborne Media, 2013.

[12] Bin He and Yonggang Li, "Big Data Reduction and Optimization in Sensor Monitoring Network," Journal of Applied Mathematics, vol. 2014, Article ID 294591, 8 pages, 2014. https://doi.org/10.1155/2014/294591.

[13] Zheng, Kan, Zhe Yang, Kuan Zhang, Periklis Chatzimisios, Kan Yang, and Wei Xiang. "Big data-driven optimization for mobile networks toward 5G." IEEE network 30, no. 1 (2016): 44-51.

[14] Thiry, Laurent, Heng Zhao, and Michel Hassenforder. "Categories for (Big) Data models and optimization." Journal of Big Data 5, no. 1 (2018): 21.

[15] El Majdouli, Mohamed Amine, Ismail Rbouh, Saad Bougrine, Bouazza El Benani, and Abdelhakim Ameur El Imrani. "Fireworks algorithm framework for Big Data optimization." Memetic Computing 8, no. 4 (2016): 333-347.

[16] Mateen, Ahmed, and Kashif Ali. "Optimization strategies through big-data migration in distributed cloud databases." In 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), pp. 96-99. IEEE, 2017.

[17] Wang, Yuzhu, Huiqun Hao, Junqiang Zhang, Jinrong Jiang, Juanxiong He, and Yan Ma. "Performance optimization and evaluation for parallel processing of big data in earth system models." Cluster Computing (2017): 1-11.

[18] Kusiak, Andrew. "Smart manufacturing must embrace big data." Nature 544, no. 7648 (2017): 23-25.

[19] Chung, I-Hsin, Tara N. Sainath, Bhuvana Ramabhadran, Michael Picheny, John Gunnels, Vernon Austel, Upendra Chauhari, and Brian Kingsbury. "Parallel deep neural network training for big data on blue gene/q." IEEE Transactions on Parallel and Distributed Systems 28, no. 6 (2016): 1703-1714.

[20] Sreedhar, Dheeraj, Vaibhav Saxena, Yogish Sabharwal, Ashish Verma, and Sameer Kumar. "Efficient Training of Convolutional Neural Nets on Large Distributed Systems." In 2018 IEEE International Conference on Cluster Computing (CLUSTER), pp. 392-401. IEEE, 2018.

[21] Cevher, Volkan, Stephen Becker, and Mark Schmidt. "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics." IEEE Signal Processing Magazine 31, no. 5 (2014): 32-43.

[22] Notarnicola, Ivano, and Giuseppe Notarstefano. "A randomized primal distributed algorithm for partitioned and big-data non-convex optimization." In 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 153-158. IEEE, 2016.

[23] Notarnicola, Ivano, Ruggero Carli, and Giuseppe Notarstefano. "Distributed partitioned big-data optimization via asynchronous dual decomposition." IEEE Transactions on Control of Network Systems 5, no. 4 (2017): 1910-1919.

[24] Notarnicola, Ivano, Ying Sun, Gesualdo Scutari, and Giuseppe Notarstefano. "Distributed big-data optimization via block communications." In 2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pp. 1-5. IEEE, 2017.

[25] Notarnicola, Ivano, Ying Sun, Gesualdo Scutari, and Giuseppe Notarstefano. "Distributed big-data optimization via block-iterative convexification and averaging." In 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 2281-2288. IEEE, 2017.

[26] Liu, Lanchao, and Zhu Han. "Multi-block ADMM for big data optimization in modern communication networks." arXiv preprint arXiv:1504.01809 (2015).

[27] Richtárik, Peter, and Martin Takáč. "Parallel coordinate descent methods for big data optimization." Mathematical Programming 156, no. 1-2 (2016): 433-484.

[28] Chen, Siguang, Kun Wang, Chuanxin Zhao, Haijun Zhang, and Yanfei Sun. "Accelerated distributed optimization design for reconstruction of big sensory data." IEEE Internet of Things Journal 4, no. 5 (2017): 1716-1725.

[29] Ning, Chao, and Fengqi You. "Data-driven stochastic robust optimization: General computational framework and algorithm leveraging machine learning for optimization under uncertainty in the big data era." Computers & Chemical Engineering 111 (2018): 115-133.

[30] Chi, Yuejie. "Low-Rank Matrix Completion [Lecture Notes]." IEEE Signal Processing Magazine 35, no. 5 (2018): 178-181.

[31] Ahmad, Awais, Murad Khan, Anand Paul, Sadia Din, M. Mazhar Rathore, Gwanggil Jeon, and Gyu Sang Choi. "Toward modeling and optimization of features selection in Big Data based social Internet of Things." Future Generation Computer Systems 82 (2018): 715-726.

[32] Elsayed, Saber, and Ruhul Sarker. "Differential evolution framework for big data optimization." Memetic Computing 8, no. 1 (2016): 17-33.

[33] Zheng, Zijie, Lingyang Song, Zhu Han, Geoffrey Ye Li, and H. Vincent Poor. "Game Theory for Big Data Processing: Multileader Multifollower Game-Based ADMM." IEEE Transactions on Signal Processing 66, no. 15 (2018): 3933-3945.

[34] Yi, Jiao-Hong, Suash Deb, Junyu Dong, Amir H. Alavi, and Gai-Ge Wang. "An improved NSGA-III Algorithm with adaptive mutation operator for big data optimization problems." Future Generation Computer Systems 88 (2018): 571-585.

[35] Barba-González, Cristóbal, José García-Nieto, Antonio J. Nebro, José A. Cordero, Juan J. Durillo, Ismael Navas-Delgado, and José F. Aldana-Montes. "jMetalSP: a framework for dynamic multi-objective big data optimization." Applied Soft Computing 69 (2018): 737-748.

[36] Povoda, Lukas, Radim Burget, Malay Kishore Dutta, and Namita Sengar. "Genetic optimization of big data sentiment analysis." In 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), pp. 141-144. IEEE, 2017.

[37] Sabar, Nasser R., Jemal Abawajy, and John Yearwood. "Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems." IEEE Transactions on Evolutionary Computation 21, no. 2 (2016): 315-327.

[38] Slavakis, Konstantinos, Georgios B. Giannakis, and Gonzalo Mateos. "Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge." IEEE Signal Processing Magazine 31, no. 5 (2014): 18-31.

[39] Patel, Aditya B., Manashvi Birla, and Ushma Nair. "Addressing big data problem using Hadoop and Map Reduce." In 2012 Nirma University International Conference on Engineering (NUiCONE), pp. 1-5. IEEE, 2012.

[40] Li, Chaojie, Xinghuo Yu, Tingwen Huang, and Xing He. "Distributed optimal consensus over resource allocation network and its application to dynamical economic dispatch." IEEE transactions on neural networks and learning systems 29, no. 6 (2017): 2407-2418.

[41] Kim, Sunyoung, and Masakazu Kojima. "Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations." Computational Optimization and Applications 26, no. 2 (2003): 143-154.

[42] Sayed, Ali H. Fundamentals of adaptive filtering. John Wiley & Sons, 2003.

[43] Sayed, Ali H. "Diffusion adaptation over networks." In Academic Press Library in Signal Processing, vol. 3, pp. 323-453. Elsevier, 2014.

[44] Qiu, Robert, and Michael Wicks. Cognitive networked sensing and big data. Springer New York, 2014.

[45] Hong, Mingyi, Meisam Razaviyayn, Zhi-Quan Luo, and Jong-Shi Pang. "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing." IEEE Signal Processing Magazine 33, no. 1 (2015): 57-77.

[46] Facchinei, Francisco, Gesualdo Scutari, and Simone Sagratella. "Parallel selective algorithms for nonconvex big data optimization." IEEE Transactions on Signal Processing 63, no. 7 (2015): 1874-1889. [47] Daneshmand, Amir, Francisco Facchinei, Vyacheslav Kungurtsev, and Gesualdo Scutari. "Hybrid random/deterministic parallel algorithms for convex and nonconvex big data optimization." IEEE Transactions on Signal Processing 63, no. 15 (2015): 3914-3929.

[48] Scott, Steven L., Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. "Bayes and big data: The consensus Monte Carlo algorithm." International Journal of Management Science and Engineering Management 11, no. 2 (2016): 78-88.

[49] Sayed, Ali H. "Adaptation, learning, and optimization over networks." Foundations and Trends® in Machine Learning 7, no. 4-5 (2014): 311-801.

[50] Boyd, Stephen, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers." Foundations and Trends® in Machine learning 3, no. 1 (2011): 1-122.

[51] Mohammadi, Mehdi, Ala Al-Fuqaha, Sameh Sorour, and Mohsen Guizani. "Deep learning for IoT big data and streaming analytics: A survey." IEEE Communications Surveys & Tutorials 20, no. 4 (2018): 2923-2960.

[52] Zhang, Ning, Peng Yang, Ju Ren, Dajiang Chen, Li Yu, and Xuemin Shen. "Synergy of big data and 5g wireless networks: opportunities, approaches, and challenges." IEEE Wireless Communications 25, no. 1 (2018): 12-18.

[53] Koning, Ruud H., Heinz Neudecker, and Tom Wansbeek. "Block Kronecker products and the vecb operator." Linear algebra and its applications 149 (1991): 165-184.

[54] Cattivelli, Federico S., and Ali H. Sayed. "Diffusion LMS strategies for distributed estimation." IEEE Transactions on Signal Processing 58, no. 3 (2009): 1035-1048.

[55] Fernandez-Bes, Jesus, Jerónimo Arenas-García, and Ali H. Sayed. "Adjustment of combination weights over adaptive diffusion networks." In 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6409-6413. IEEE, 2014.

[56] Wei, Jieqiang, Alexander Johansson, Henrik Sandberg, Karl H. Johansson, and Jie Chen. "Optimal weight allocation of dynamic distribution networks and positive semi-definiteness of signed Laplacians." arXiv preprint arXiv:1803.05640 (2018).

[57] Cassano, Lucas, Kun Yuan, and Ali H. Sayed. "Distributed value-function learning with linear convergence rates." In 2019 18th European Control Conference (ECC), pp. 505-511. IEEE, 2019.

[58] Chen, Jie, Cédric Richard, and Ali H. Sayed. "Diffusion LMS over multitask networks." IEEE Transactions on Signal Processing 63, no. 11 (2015): 2733-2748.

6 Appendix

6.1 Simulation Codes

Here we present some part of our simulation codes. This research is continuing; therefore, the main components of simulation codes that might be used for future works are protected.

[Spectrum Sensing on Cognitive Radio Using Consensus]

[Part 1: Generate data for the cognitive network]

```
clear all;
close all;
clc;
u=1000;%time bandwidth factor
N=2*u;%samples
a=2;%path loss exponent
C=2;%constant losses
Crs=15; %Number of cognitive radio users
PdAnd=0;
%----------Pfa------------%
Pf=0.01:0.01:1;
Pfa=Pf.^2;
%---------signal-----%
t=1:N;
s1 = cos(pi*t);
stem(t,s1)
s1power=var(s1);
%-------- SNR ----------%
% Snrdb=-15:1:15;
Snrdb=15;
Snreal=power(10,Snrdb/10);%Linear Snr


% while Snrdb<15%
  lamda=ones(1,100);
```

```
for i=1:length(Pfa)
lamda(i)=gammaincinv(1-Pfa(i),u)*2; %theshold
% lamdadB=10*log10(lamda);
%--------Local spectrum sensing---------%
d=ones(1,10);
for j=1:Crs %for each node
detect=0;
d(j)=7+1.1*rand(); %random distanse
PL=C*(d(j)^-a); %path loss
for sim=1:10%Monte Carlo Simulation for 100 noise realisation
%-------------AWGN channel--------------------%
noise = randn(1,N); %Noise production with zero mean and s^2 var
noise_power = mean(noise.^2); %noise average power
amp = sqrt(noise.^2*Snreal);
s1=amp.*s1./abs(s1);
% SNRdB_Sample=10*log10(s1.^2./(noise.^2));
Rec_signal=s1+noise;%received signal
localSNR=ones(1,10);
localSNR(j)=mean(abs(s1).^2)*PL/noise_power;%local snr
pdth=cell(10,100);
pdth{1,100}=1:100;
pdth{10,10}='string';
Pdth(j,i)=marcumq(sqrt(2*localSNR(j)),sqrt(lamda(i )),u);%Pd for j node
%Computation of Test statistic for energy detection
Sum=abs(Rec_signal).^2*PL;
Test=ones(1,10);
Test(j,sim)=sum(Sum)
if (Test(j,sim)>lamda(i))
detect=detect+1;
end
```

```
end %END Monte Carlo
Pdsim=ones(1,10);
Pdsim(j)=detect/sim; %Pd of simulation for the j-th CRuser
end
PdAND=ones(1,100);
PdAND(i)=prod(Pdsim);
PdOR=ones(1,100);
PdOR(i)=1-prod(1-Pdsim);
end
PdAND5=(Pdth(5,:)).^5;
Pmd5=1-PdAND5;
PdANDth=(Pdth(Crs,:)).^Crs;
PmdANDth=1-PdANDth; %Probability of miss detection
Pmdsim=1-PdAND;
```

[part 2: consensus strategy]

```
clc        % clear command window
clear all  % clear all variables
close all  % close all figures
h = 2; % h=2 for complex data
N = 2; % number of cognitive users
M = 3; % number of taps per user (defines collaboration)
Num_iter = 75;   % number of iterations per trial
Num_trial = 300;  % number of trials
mu_max = 0.00001; % uniform step size across the network
mu = mu_max*ones(N,1); % uniform step-sizes
A = [0.2 0.8; 0.8 0.2];
p =(1/N)*ones(N,1);      % Perron eigenvector
% set the power level for each random quantity
sigma_u2 = 0.5/mu_max;
sigma_v2 = 0.05*ones(N,1);
```

```
RU = sigma_u2*ones(M,N);
sqRU = sqrt(sigma_u2)*ones(M,N);
% generate wo
wo = randn(M,1)+1j*randn(M,1);
wo = wo / norm(wo,2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generating the signal and noise powers
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if h == 1
  wo = randn(M,1);  % real data
  wo = wo / norm(wo,2);
else
  wo = randn(M,1)+1j*randn(M,1); % complex data
  wo = wo / norm(wo,2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Running the experiments to generate the learning curves
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSD_av_CON    = zeros(1,Num_iter);   % average MSD curve for the consensus network.
MSD_agent_CON = zeros(N,Num_iter);    % each row contains the MSD curve for the corresponding cognitive users in the consensus network
wb = waitbar(0,'Simulating...Please wait');
for L=1:Num_trial % iterating over experiments
   % consensus initialization
   psi_CON = zeros(M,N);       % psi column vectors for all cognitive users in the consensus network
   w_CON   = zeros(M,N);       % iterate column vectors for all cognitive users in the consensus network
   tilde_w_CON  = zeros(M,N);  % error column vectors for all cognitive users in the consensus network
   for i=1:Num_iter  % iterating over time
      waitbar(((L-1)*Num_iter+i)/(Num_iter*Num_trial),wb);
      for k=1:N % consensus
```

```
        psi_CON(:,k) = zeros(M,1);
        for l=1:N
          psi_CON(:,k) = psi_CON(:,k) + A(l,k)*w_CON(:,l); % consensus (consultation step)
        end
      end
      for k=1:N % generate data for each cognitive users at time i
        if h==1 % real data
          uk = randn(1,M)*diag(sqRU(:,k)); % Gaussian row regression vector
          dk = uk*wo + randn*sqrt(sigma_v2(k));
        else
          uk = complexrandn(1,M)*diag(sqRU(:,k)); % Gaussian row regression vector
          dk = uk*wo + complexrandn(1,1)*sqrt(sigma_v2(k));
        end
        w_CON(:,k)  = psi_CON(:,k) + (2/h)*mu(k)*uk'*(dk - uk*w_CON(:,k));   % consensus (adaptation step)
      end
      for k=1:N
        w(:,k) = zeros(M,1);
        tilde_w_CON(:,k) = wo - w_CON(:,k); % consensus
        MSD_agent_CON(k,i) = MSD_agent_CON(k,i) + (norm(tilde_w_CON(:,k),2))^2;
      end
    end
  end
  % consensus network learning curve
  MSD_agent_CON = MSD_agent_CON/Num_trial; % each row contains the MSD evolution of the corresponding cognitive user
  MSD_av_CON = sum(MSD_agent_CON)/N;       % average MSD evolution of the network
  MSD_av_db_CON = 10*log10(MSD_av_CON);    % dB
```

[Optimal Weightning]

[Part 1: network Generation]

```
function [Adjacency,Laplacian,Algebraic_Connectivity,Degree_Vector,Coordinates] = generate_topology(Num_nodes,Type,Parameter)

N = Num_nodes;  % Number of nodes.

A = zeros(N,N); % Adjacency matrix.

L = zeros(N,N); % Laplacian matrix.

if Type == 1

    r = Parameter;  % Nodes within this radius from each other are declared to be neighbors.

else

    p = Parameter;  % Nodes k and l are declated neighbors according to a binomial distribution with probability p.

end

% We first generate N random (x,y) coordinates in the square region [0,1.2]x[0,1.2]

x_coordinates = rand(1,N) + 0.1;

y_coordinates = rand(1,N) + 0.1;

Coordinates = [x_coordinates' y_coordinates'];

if Type == 1 % distance criterion

  for k=1:N

    for l=1:N

        d = sqrt((x_coordinates(1,k)-x_coordinates(1,l))^2 + (y_coordinates(1,k)-y_coordinates(1,l))^2);

        if d <= r

            A(k,l) = 1; % set entry in adjacency matrix to one if nodes k and l should be neighbors.

        end

    end

  end

end

if Type == 2 % binomial criterion

  for k=1:N

      A(k,k) = 1; % a node is always connected to itself in this construction

    for l=k+1:N
```

```
      b = rand; % generate a uniform random variable in the interval [0,1]
      if b <= p % if b falls within the interval [0,p], then we connect the nodes (emulating a binomial variable with prob. p)
        A(k,l) = 1; % set entry in adjacency matrix to one if nodes k and l should be neighbors.
        A(l,k) = 1;
      end
    end
  end
end

Adjacency = A; % adjacency matrix.
% We determine the number of neighbors of each node from the adjacency matrix
num_nb = zeros(N,1);
for k=1:N
   num_nb(k) = sum(A(k,:));
end
Degree_Vector = num_nb;  % vector of degrees for the various nodes
for k=1:N
   L(k,k) = max(0, sum(A(k,:))-1); % set diagonal entry to zero if degree-1 for node k is negative.
   for l=k+1:N
      L(k,l) = -1*A(k,l);
      L(l,k) = -1*A(l,k);
   end
end
sigma = svd(L); % vector of singular values of L.
Laplacian = L; %Laplacian matrix
Algebraic_Connectivity = sigma(N-1); % algebraic connectivity
if sigma(N-1) < 1e-4 % checking if second smallest singular value is positive (sufficiently away from zero).
   return % network is not connection; returns to calling the function again to try a new network construction.
End
function plot_topology(Adjacency,Coordinates,Color)
```

```
A = Adjacency;      % adjacency matrix
N = max(size(A));   % number of agents
x_coordinates = Coordinates(:,1);
y_coordinates = Coordinates(:,2);
figure
hold on
for k=1:N
  for l=1:N
    if A(k,l)>0
      plot([x_coordinates(k),x_coordinates(l)],[y_coordinates(k),y_coordinates(l)],'b-','LineWidth',1.5);
    end
  end
end
for k=1:N
 if Color(k) == 0
     plot(x_coordinates(k),y_coordinates(k),'o','MarkerEdgeColor','b','MarkerFaceColor','y','MarkerSize',10);
 else
   if Color(k) == 1
      plot(x_coordinates(k),y_coordinates(k),'o','MarkerEdgeColor','b','MarkerFaceColor','r','MarkerSize',10);
   else % green
      plot(x_coordinates(k),y_coordinates(k),'o','MarkerEdgeColor','b','MarkerFaceColor','g','MarkerSize',10);
   end
 end
end
axis([0,1.2,0,1.2]);
axis square
grid
for k=1:N
  text(x_coordinates(k)+0.03,y_coordinates(k)+0.03,num2str(k),'Fontsize',7);
end
```

[policy rules]

Only parts of the code is presented.

```
clc          % clear command window
clear all  % clear all variables
close all  % close all figures
h = 1;                % use h=1 for real data; use h=2 for complex data
uniform_step = 1; % set it to 0 for different step-sizes across all agents; set it to 1 for uniform step-sizes: \mu_k=\mu, zeta_k=\zeta
uniform_reg  = 1; % set it to the following values:
mu_max   = 0.001;    % maximum step-size parameter used by the agents for adapting the weight estimates
mu_max_adap = 0.001; % maximum step-size for adaptive rule since adaptation of coefficients slows down convergence; allows separate control if desired
zeta_max = 0.01;       % maximum step-size parameter used by the agents to adapt their combination weights; used only for adaptive combination policies
N = 20;               % number of agents in the network
M = 10;               % size of wo, which is Mx1
Num_iter  = 5000;  % number of iterations per experiment
switch_time = 1000; % time at which to switch from uniform to adaptive rule in one of the simulations
Num_trial =  100;   % number of experiments
Gamma = zeros(N,N);  % matrix containing the parameters gamma_{\ell k}^2, which are used only in the adaptive combination implementation
type_topology = 1; % use 1 for neighbors to be decided by closeness in distance; use 2 for neighbors to be decided randomly using binary decisions
radius = 0.3;      % size of radius to be used to determine which nodes are neighbors for type_topology = 1.
prob   = 0.2;      % probability value to be used to determine which nodes are neighbors for type_topology = 2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generating the signal and noise powers
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if h == 1 % wo is used in the same model case and also for centralized and non-cooperative implementations.
    wo = randn(M,1);  % real data (vector)
    wo = wo / norm(wo,2);
else
    wo = randn(M,1)+1j*randn(M,1); % complex data (vector)
    wo = wo / norm(wo,2);
end
if uniform_step == 0
    mu   = mu_max*(0.5*rand(N,1)+0.5);    % generates the step-sizes mu_k for all agents by multiplying mu_max by random numbers between 0.5 and 1
    zeta = zeta_max*(0.5*rand(N,1)+0.5);  % generates the step-sizes zeta_k for all agents by multiplying zeta_max by random numbers between 0.5 and 1
    mu_adap   = mu_max_adap*(0.5*rand(N,1)+0.5); % for adaptive rule

else
    mu  = mu_max*ones(N,1);   % use this line if it is desired to set all step-sizes to the same value across the agents.
    mu_adap = mu_max_adap*ones(N,1);   % use this line if it is desired to set all step-sizes to the same value across the agents.
    zeta = zeta_max*ones(N,1); % use this line if it is desired to set all step-sizes to the same value across the agents.
end
```

```
for k=1:N
  b_relvar(k)  = mu(k)*mu(k)*sigma_v2(k)*trace_values(k);
  b_relvar(k)  = 1/b_relvar(k);
end
[A_uniform, p_uniform]       = generate_combination_policy(adjacency,ones(N,1),'uniform');
[A_metropolis, p_metropolis] = generate_combination_policy(adjacency,ones(N,1),'metropolis');
[A_relvar, p_relvar]         = generate_combination_policy(adjacency,b_relvar,'relvar');
w_star = wo;
%-------------------------------------------------------------
```