Distributed Processing of Blind Source Separation

by

Seyed Reza Mir Alavi

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Engineering and Computer Science.

Victoria University of Wellington
2017

Abstract

Communication is performed by transmitting signals through a medium. It is common that signals originating from different sources are mixed in the transport medium. The operation of separating source signals without prior information about the sources is referred to as blind source separation (BSS). Blind source separation for wireless sensor networks has recently received attention because of low cost and the easy coverage of large areas. Distributed processing is attractive as it is scalable and consumes low power. Existing distributed BSS algorithms either require a fully connected pattern of connectivity, to ensure the good performance, or require a high computational load at each sensor node, to enhance the scalability. This motivates us to develop distributed BSS algorithms that can be implemented over any arbitrary graph with fully shared computations and with good performance.

This thesis presents three studies on distributed algorithms. The first two studies are on existing distributed algorithms that are used in linearly constrained convex optimization problems, which are common in signal processing and machine learning. The studies are aimed at improving the algorithms in terms of computational complexity, communication cost, processors coordination and scalability. This makes them more suitable for implementation on sensor networks, thus forming a basis for the development of distributed BSS algorithms on sensor networks in our third study.

In the first study, we consider constrained problems in which the constraint includes a weighted sum of all the decision variables. By formulating a constrained dual problem associated to the original constrained problem, we were able to develop a distributed algorithm that can be run both synchronously and asynchronously on any arbitrary graph with lower communication cost than traditional distributed algorithms.

In the second study, we consider constrained problems in which the constraint is separable. By making use of the augmented Lagrangian function and splitting

the dual variable (Lagrange multiplier) associated to each partial constraint, we were able to develop a distributed *fully* asynchronous algorithm with lower computational complexity than traditional distributed algorithms. The simplicity of the algorithm is the consequence of approximating the constraint on the equality of the decoupled dual variables. We also provide a measure of the inaccuracy in such an approximation on the optimal value of the primal objective function.
Finally, in the third study, we investigate distributed processing solutions for BSS on sensor networks. We propose two distributed processing schemes for BSS that we refer to as scheme 1 and scheme 2. In scheme 1, each sensor node estimates one specific source signal while in scheme 2, by formulating a consensus optimization problem, each sensor node estimates all source signals in a fully shared computation manner. Our proposed algorithms carry the following features: low computational complexity, low power consumption, low data transmission rate, scalability and excellent performance over arbitrary graphs. Although all of our proposed algorithms share the aforementioned properties, each of them is superior in one or some of the features compared to the others. Comparative experimental results show that among all our proposed distributed BSS algorithms, a variant of scheme 1 performs best when all features are considered. This is achieved by making use of the concept of *pairwise* mutual information along with adding a sparsity assumption on the parameters of the model that is used in BSS.

Acknowledgments

This PhD has consistently challenged me over the last three years. Although the process of fulfilling all the requirements had stretched me to my limits, I would not have been able to complete it without the assistance of a few very special people. I would like to express my sincere gratitude to my supervisor, Prof. Bastiaan Kleijn, for his unwavering support and much needed words of wisdom over the years.
This would not have been possible without the encouragement and support from my loving parents. Words cannot express how grateful I am for all the sacrifices you have made on my behalf. And to my loving wife, who had to put up with all my stress and mischief, thank you, you're the greatest!

Publications of the thesis

The contents of this thesis have been published or will appear in the form of the following papers,

I S. R. Mir Alavi, W. B. Kleijn, "Distributed linear blind source separation over wireless sensor networks with arbitrary connectivity patterns", in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016.

II S. R. Mir Alavi, W. B. Kleijn, "Communication efficient asynchronous distributed linearly constrained convex optimization with $O(1/k)$ rate of convergence“. *Submitted for publication.*

III S. R. Mir Alavi, W. B. Kleijn, "Fully asynchronous distributed optimization over graphs using a regularized augmented Lagrangian function and its application to the consensus problem", in *To be submitted for publication.*

IV S. R. Mir Alavi, W. B. Kleijn, "Distributed adaptive linear blind source separation over arbitrary graphs“. *Submitted for publication.*

Acronyms

ADMM	Alternating Direction Method of Multipliers
AMM	Augmented Method of Multipliers
ATC	Adapt Then Combine
BP	Belief Propagation
BSS	Blind Source Separation
CTA	Combine Then Adapt
DFT	Discrete Fourier Transform
EM	Expectation Maximization
FA	Factor Analysis
GaBP	Gaussian Belief Propagation
GLiCD	Generalized Linear Coordinate Descent
ICA	Independent Component Analysis

ID	Identification
LiCD	Linear Coordinate Descent
LMS	Least Mean Square
MAP	Maximum A Posteriori
ML	Maximum Likelihood
MMI	Minimum Mutual Information
MSD	Mean Square Deviation
MSE	Mean Squared Error
NFA	Non-linear Factor Analysis
NICA	Non-linear Independent Component Analysis
NLMS	Normalized Least Mean Square
NLPCA	Non-Linear Principle Component Analysis
NMSE	Normalized Mean Squared Error
PCA	Principle Component Analysis
PGMs	Probabilistic Graphical Models
RLS	Recursive Least Square
SIR	Signal to Interference Ratio

SNR	Signal to Noise Ratio
WSN	Wireless Sensor Network

Contents

1 Introduction

1.1 Motivation and approaches

Radio and acoustic signals play a significant role in communication. It is natural that signals transmitted from independent sources are mixed during transmission before they are observed. As each source signal typically carries its own information independent of the other source signals, it is common that one needs to extract the original source signals from their mixture. Blind source separation (BSS) is a strategy that can be used for this purpose, e.g., [38, 198].

The term "blind" in BSS refers to an operation without or with poor information about the source signals and the mixing process. However, to obtain satisfactory results, one must consider an appropriate model for the mixing process. For example, in acoustics the input-output data relationship in a time-invariant reverberant environment requires a convolutive model while a much simpler multiplicative model is satisfactory in a delay-restricted environment. The proposed solutions for the multiplicative models, known as *linear* BSS algorithms, can also be applied to convolutive mixtures by transferring the observed data to the frequency domain via Fourier Transform which results in a multiplicative model for each frequency bin [89, 128, 147, 171].

Blind source separation has many applications in audio signal processing, for example [184]. It has been also widely used in image processing [18, 117], biomedical [122, 183], and finance [16, 119]. An example application of BSS in audio signal processing is in the cocktail party problem where the objective is to focus on a single conversation in an environment with multiple overlapping voices.

BSS algorithms can be divided into *batch* algorithms and *adaptive* algorithms. Both approaches are generally iterative methods. Batch algorithms use the entire observed data for optimization in each iteration, while in adaptive algorithms only a single observation (current observed data) is used, using the stochastic gradient method. This property of adaptive algorithms makes them suitable for real-time applications and in scenarios where the underlying sources are allowed to be mobile, albeit with slow rate of movement.

Many approaches have been introduced in the literature to address BSS. One of the well-known approaches is the independent component analysis (ICA). In ICA different objective functions in terms of the parameters of the model are formulated. This is motivated by applying optimization techniques to the functions to estimate the parameters as well as the sources. The objective functions are generally categorized into functions that measure the dependency or non-gaussianity of the estimated source signals. Examples of those are kurtosis [143], negentropy [84], mutual information [11] and likelihood function [113]. The quality of separation can be different depending on what objective function and assumptions is used [99, 102, 134].

To perform the blind source separation a set of different mixture signals must be recorded. This can be achieved by a set of sensors (microphones) at different spatial locations. Wireless sensor networks (WSN) [3, 4, 182, 196] are a natural, low-cost means for capturing the signals. In contrast to wired networks of sensors, WSNs facilitate a good coverage of an area where signals of interest are present at a low deployment cost.

Algorithms for blind source separation of signals acquired over sensor networks can be based on centralized processing or on de-centralized processing. In the centralized processing approach, e.g., [44, 157], the recordings of all sensors are sent to a centralized processing unit that performs the separation procedure. Limitations of the centralized approach include the need for powerful processing hardware, high power consumption of the transmission, and lack of scalability

[63]. It may be even impossible to apply centralized approach due to the high physical length of the links and interference.

In contrast to the centralized approaches, the de-centralized, in-network processing approaches have the potential to be scalable and have a low power consumption of the transmission [129,154]. In fact the de-centralized processing is a more realistic approach for real-world large scale problems.

A simple form of de-centralized BSS was discussed in [45, 123], where the separation procedure is pararallelized over a number of processors. However, the methods in [45, 123] retain the need to aggregate the observations of all sensors in each individual node. This imposes the need for a full connectivity pattern over the graph of the network and, hence, restricts the scalability of the BSS algorithm. A more scalable distributed algorithm is introduced in [79], but at the cost of a significantly higher computational load at each sensor node.

Regardless of the lack of research on distributed BSS, there is research on distributed algorithms for a variety of applications in signal processing, machine learning and statistics [131, 145]. In the latter fields usually a constrained or unconstrained optimization problem is formulated. The objective function and constraint would be different depending on the application.

To develop distributed algorithms for the constrained and unconstrained problems, it is assumed that the primal objective is separable (forms a sum of N functions). The separability of the objective function allows implementation of distributed algorithms on a network of N agents (processors), where each agent handles a small portion of the optimization task.

In the case of constrained problems, existing distributed algorithms rely on either the Lagrangian or the augmented Lagrangian function that is associated with the constrained optimization problem. Methods based on the Lagrangian have a convergence rate of $O(1/\sqrt{k})$, meaning that the error of estimation decreases by a factor of $1/\sqrt{k}$ where k denotes the iteration counter. For the constrained problems, in which the constraint includes a weighted sum of *all* the decision variables, Lagrangian-based methods require a communication graph with star topology, where there is a master agent to which all other agents are connected. Methods based on the augmented Lagrangian have a quicker rate of convergence, $O(1/k)$, at the cost of requiring a fully connected network.

1.2 Contributions of the thesis

In this thesis, we investigate distributed forms of the adaptive BSS algorithm. The objective of the thesis is to develop distributed algorithms that can be applied over wireless sensor networks, where each sensor node is equipped with a microphone and a microprocessor, to track the source signals in a real-time processing fashion. An example real world application of the proposed distributed BSS algorithms is in audio source separation in cocktail parties. The proposed distributed BSS algorithms are aimed to be applied over any connected network with any connectivity pattern. Applying the proposed algorithms over a well designed sparse network reduces the energy consumption for data transmission and enhances the scalability as opposed to the centralized processing approaches. An appropriate design of the network means that no transmission over long distances is required. We confirm the strong performance of the proposed distributed BSS algorithms with experiments for the case of delay-restricted environments where the observations are obtained from the linear instantaneous mixture of the sources. The method can be easily extended to apply to convolutional scenarios using conventional frequency domain approaches [89, 128, 171]. For a thorough review of the frequency domain approaches in the centralized case the reader is referred to [85, 147].

In this thesis, we also investigate efficient distributed algorithms for linearly constrained problems. Efficiency of the proposed distributed algorithms is in terms of the required communication processes, required bandwidth for a given communication channel and the computational complexity at each agent over a network of agents. The proposed distributed algorithms not only can be applied on blind source separation but also can be implemented in a variety of applications in signal and image processing such as the basis pursuit problem [27, 33], image denoising [153, 160] and the consensus problem [22, 124, 137, 168].

The most important contributions of this thesis are summarised as follows:

- Development of a communication-efficient asynchronous distributed algorithm for linearly constrained convex optimization problems, where all the primal variables are coupled by a global single block linear constraint.
- A straightforward derivation of a fully asynchronous distributed algorithm

with low computational complexity and its application to the consensus problem.

- Reducing the communication cost and channel bandwidth requirements in synchronous distributed optimization over wireless sensor networks.
- Development of a stable blind source separation algorithm for non-stationary signals, where the optimization is performed over the *mixing* matrix.
- Distributed processing of *sparse* blind source separation over arbitrary connected graphs with minimum power consumption of the transmission.
- Distributed processing of the blind source separation over arbitrary graphs with the performance analogous to the corresponding centralized algorithm.

1.3 Structure of the thesis

The rest of this thesis is organized as follows:
In Chapter 2, we review approaches that are proposed for the linear blind source separation. The main focus of Chapter 2 is on the linear instantaneous mixing model and investigates sophisticated methods as solution for the blind source separation problem.
In Chapter 3 we show how separability of the objective function of an optimization problem allows us to develop distributed algorithms on a network of agents. We will see that separability allows us to allocate separate tasks to each agent of the network. The main focus of this chapter is on unconstrained optimization problems.
In Chapter 4, we investigate distributed algorithms as efficient solutions for the linearly constrained problems with separable objective functions, where the constraint includes a weighted sum of all the primal variables. In this chapter we propose a distributed algorithm that can be applied over any connected graph with any connectivity pattern in contrast to the traditional algorithms. The main part of Chapter 4 is presented in publication II.
In Chapter 5, we consider the linearly constrained optimization problems, where there are more than one block of constraint. In this chapter we propose a fully

asynchronous distributed algorithm with low computational complexity and investigate its application in the consensus problem. We also derive synchronous algorithms with low channel bandwidth requirements. The main part of Chapter 5 is presented in publication III.

In Chapter 6, we propose distributed algorithms for blind source separation over sensor networks. The proposed algorithms in this chapter can be applied over any arbitrary graph. A summary of Chapter 6 is presented in publication I while the main part of it is presented in publication IV.

2 Linear blind source separation

Learning from data efficiently requires the existence of a representative model for the data a-priori. In general, any observed data in real-world can be represented either by a linear or a non-linear model. Consider a non-linear relationship between a set of source signals, $S(t)$, and their observations as, $b(t)$, in presence of additive noise, $n(t)$, as:

$$b_{N\times 1}(t) = F(S_{M\times 1}(t)) + n_{N\times 1}(t), \quad t = 1, ..., T \tag{2.1}$$

where $F(.) : \mathbb{R}^M \rightarrow \mathbb{R}^N$ is a non-linear function. Given the model in (2.1), the objective of the non-linear blind source separation problem is to learn the source vector $S(t)$, $\forall t = 1, ..., T$ given only the observed data, b, without any prior knowledge about the sources, the number of them and the function $F(.)$. However, in non-linear BSS it is usually assumed that the number of observations are greater than or equal to the number of original sources. The ambiguity about both the sources and the function $F(.)$ makes the non-linear BSS problem highly vague and makes the estimation problem highly difficult, at-least compared to the linear models. Regardless of the latter issues there are many learning methods, such as Non-Linear Principle Component Analysis (NLPCA), Non-linear Factor Analysis (NFA), Non-linear Independent Component Analysis (NICA), etc, that

has been successfully applied for non-linear BSS [101,105,159,193,195,210]. Compared to non-linear models, linear models are much simpler while they are able to capture the underlying nature of the observed data in a wide range of applications. For example in the cocktail party problem, two linear generative models can well represent the set of observed data depending on the level of reverberation in the environment of the talkers. The observed data can be well modelled with a simple linear (instantaneous) model in a delay-restricted environment while they are well modelled with a linear convolutive model in an environment with high level of reverberation. The complexity of the convolutive models can be reduced by transferring the observed data to the frequency domain via Fourier Transform and making a linear instantaneous model for each frequency bin. Therefore this thesis only reviews the classical approaches that are introduced for linear *instantaneous* blind source separation as they can be easily applied to address the cocktail party problem both in echoic and delay-restricted environments as well as many other problems in engineering.
This chapter starts with introducing the model that is used in linear *instantaneous* BSS in Section 2.1. It discusses the conventional approaches for linear *instantaneous* BSS in Section 2.2. Section 2.3 and 2.4 explains the Principal Component Analysis (PCA) and Factor Analysis (FA) respectively as approaches that deals with the second order statistics of the data. Section 2.5 describe the Independent Component Analysis (ICA) as a sophisticated approach that deals with higher order statistics of the data. Finally Section 2.6 discusses the difficulties and ambiguities that exist in ICA algorithms and BSS problem.

2.1 The basic model

The following generative model describes the input-output data relationship between a set of source signals as the input and their mixture as the output where the mixture is obtained from a linear instantaneous mixture of the sources plus possibly some additive noise:

$$b_{N\times1}(t) = A_{N\times M}S_{M\times1}(t) + n_{N\times1}(t), \quad t = 1, ..., T \tag{2.2}$$

where $b_{N\times1}(t) = [b_1(t), ..., b_N(t)]^T$, $b_i(t)$ contains the observations at the i^{th} sensor (microphone) at time index t and $S_{M\times1}(t) = [S_1(t), ..., S_M(t)]^T$, $S_i(t)$ contains the magnitudes of the i^{th} source signal at time index t, A is the mixing matrix and n contains the magnitudes of the noise signals.

In most of BSS methods, for the purpose of simplicity, the noise is omitted in (2.2) and it is assumed that the number of sensors is equal to the number of sources, $N = M$, so (2.2) reduces to:

$$b(t) = \sum_{i=1}^{N} A^i S_i(t), \qquad t = 1, ..., T \tag{2.3}$$

where A^i is the i^{th} column of the matrix A.

Considering the equation (2.3), the objective of linear blind source separation methods is to make an estimation over the elements of the matrix $S_{M\times T}$ without or with poor information about the matrix A and by relying only on the observations on the matrix $b_{N\times T}$.

2.2 Classical approaches

There have been proposed many approaches in the literature as solutions for linear instantaneous blind source separation. In overall, they can be divided into two categories. The first category includes approaches that use the second order statistics of the data, e.g. variance, and try to decorrelate the observed data as a solution for separation. Example methods of the first category are principal component analysis [1, 81, 93, 146, 190] and factor analysis [66, 100, 177]. The second category includes approaches that use the higher order statistics of the data, e.g. third or forth moments, and try to maximise the independency over the set of observed data as a solution for separation. Independent component analysis is an approach that sits in the second category. There are many ICA algorithms proposed in the literature. A thorough review of the ICA algorithms is presented in [35,69,87,107].

The independency in the ICA can be defined from different perspectives. According to the knowledge extracted from the central limit theorem the independency of a set of random variables is measured as a distance to normality. The more the non-Gaussian a variable is the more independent it is assumed, since it can not be represented as a weighted sum of other independent variables. The absolute value of kurtosis and the negative differential entropy (neg-entropy) are two objective functions that are used for this purpose. Example algorithms that use the kurtosis and the neg-entropy are presented in [84, 106, 118, 139]. From the information-theoretic view point, the average mutual information of a set of random variables is a measure to their mutual dependence. So the minimization of average mutual information can leads to the independency. Example algorithms that use the average mutual information as the objective function are presented in [78, 150, 194]. An alternative approach to minimization of average mutual information is based on a maximum-likelihood estimation. In the maximum-likelihood based blind source separation, the objective is to maximize the likelihood of the ICA model where a prior independency is considered over the sources. The latter independency is defined by representing the joint distribution of the sources as the product of their marginal distributions. Example ICA algorithms based on maximum-likelihood approach are presented in [109, 142, 151]. Another objective function that can be used for blind source separation as a measure of independency is derived from the neural network viewpoint [20]. In this approach, the independent sources are estimated by maximizing the output entropy of a separation neural network. The principle of maximizing the output entropy of a neural network, information maximization or infomax, is equivalent to the maximum-likelihood estimation if an appropriate activation function is used for activating the neurons of the network. The connection between infomax and maximum-likelihood is determined in [29, 149].

In problems in which the original sources are inherently independent, ICA is the best candidate for source separation in comparison with the methods that only assume a prior decorrelation about the sources, e.g. PCA and FA. For example in the cocktail party problem, where the underlying sources are human speech signals, assuming a prior independency about the sources is realistic since the process of speech production in different people is independent of each other. In fact not only the human speech signals but also many signals that are produced by different individual sources tend to have independent structure. Therefore in this thesis we aim to apply ICA as a technique for blind source separation. Not only from our perspective but it has also been widely used as a method for BSS since 1980s.

2.3 Principal component analysis

Principal component analysis was first introduced by Karl Pearson in 1901. It is a statistical approach that uses an orthogonal transformation to convert a set of correlated data into a set of uncorrelated data known as principal components. As independency of a set of data implies the uncorrelatedness, the reverse relation does not necessarily hold, PCA was used as a method for separation. PCA uses the following steps for de-correlation:
Step 1: Centering the data as:

$$b_c = b - \mu_b \tag{2.4}$$

where $\mu_b = [\mu_{b,1}, ..., \mu_{b,N}]^T$, $\mu_{b,i} = \frac{1}{T}\sum_{t=1}^{T} b_i(t)$.
Step 2: Orthogonal transformation as:

$$Y_P = V^T b_c \tag{2.5}$$

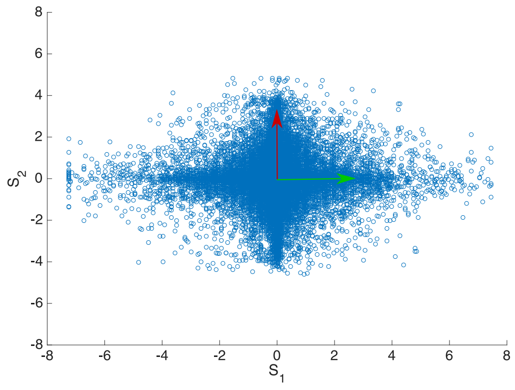
where V is a matrix that is made by stacking all the eigenvectors of the covariance matrix, $C_{b_c} = \frac{1}{T} b_c b_c^T$, of the centered data and Y_P contains the de-correlated data. It can be easily verified from the following equality that the data in Y_P are decorrelated:

$$\frac{1}{T} Y_P Y_P^T = \frac{1}{T} V^T b_c b_c^T V = V^T C_{b_c} V = V^T V D V^T V = D \tag{2.6}$$

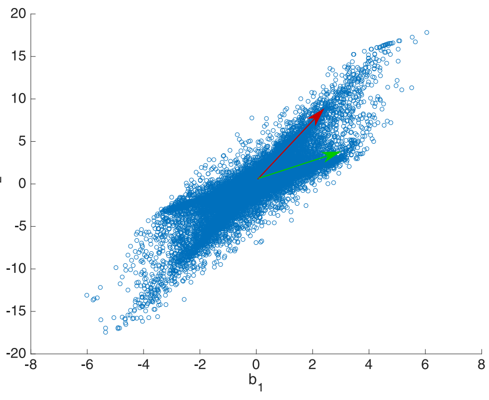
where D is a diagonal matrix that contains the eigenvalues of the covariance matrix, C_{b_c}, and $V^T = V^{-1}$ as it is an orthogonal matrix. The diagonality of D implies that the data in Y_P have pairwise zero covariance.

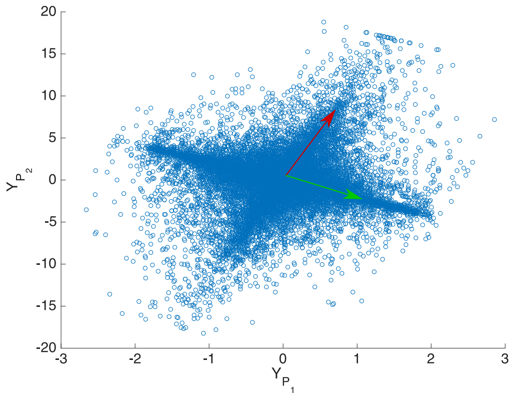
PCA can also be used for dimensionality reduction by considering only k eigenvectors, where $k < N$, that correspond to the k largest eigenvalues of the covariance matrix in equation (2.5).

Fig. 2.1 visualises the effect of PCA over a set of mixed data where the underlying sources have Laplacian distributions. The results in Fig. 2.1 were obtained by using 2 original sources (human speech signals) with 44000 samples. From Fig. 2.1 we see the lack of an additional rotation for extracting the original distributions. However, this is not the case for symmetric Gaussian distributions as they are not distinguishable with different rotations. Therefore PCA can only



(a)





(b) (c)

Figure 2.1: Visualisation of the scatter plot of the data. a) The scatter plot of the original sources. b) The scatter plot of the mixed (observed) data. c) The scatter plot of decorrelated data using PCA.

be meaningful in the sense of separation if the underlying sources have Gaussian distributions. The aforementioned limitation of the principal component analysis makes it less practical for blind source separation. However, PCA can facilitate the separation procedure by reducing redundancy over the set of observed data as a preprocessing technique for independent component analysis (ICA) algorithms. It is good to notice the connection between PCA and whitening. A set of zero mean data are said to be white if their covariance matrix is equal to identity matrix. The whitening is performed over a set of zero mean data through the following transformation:

$$Y_W = V^T D^{\frac{-1}{2}} V b_c \tag{2.7}$$

where Y_W contains the whitened data that can be verified from the following equality:

$$\begin{aligned}\frac{1}{T}Y_W Y_W^T &= \frac{1}{T}VD^{\frac{-1}{2}}V^T b_c b_c^T VD^{\frac{-1}{2}}V^T = VD^{\frac{-1}{2}}V^T C_{b_c} VD^{\frac{-1}{2}}V^T \\ &= VD^{\frac{-1}{2}}V^T VDV^T VD^{\frac{-1}{2}}V^T = I\end{aligned} \tag{2.8}$$

Similar to PCA, whitening can also be used for decorrelation. In contrast to PCA, whitening produces decorrelated data with normalized variances.

2.4 Factor analysis

Factor analysis is a method that can describe a set of possibly correlated variables in terms of potentially lower number of uncorrelated factors. There is a strong relationship between factor analysis and principal component analysis but they are not necessarily the same. FA uses different model assumptions compared to PCA. In factor analysis the model in (2.2) is used and it is also assumed that the factors S(t) have a Gaussian distribution with identity covariance matrix as:

$$S(t) \sim \mathcal{N}(0, I) \tag{2.9}$$

where $\mathcal{N}(0, I)$ denotes a normal distribution with zero mean and identity covariance matrix. The identity covariance matrix implies that the factors are decorrelated. Using (2.9) in the model (2.2) results in a distribution for the observed data as:

$$b(t) \sim \mathcal{N}(AS(t) + \mu_n, \Sigma_n) \tag{2.10}$$

where μ_n and Σ_n are the mean and covariance of the noise respectively and Σ_n is a diagonal matrix.

By denoting the parameters of the model, (A,μ_n,Σ_n), with θ we can formulate a likelihood for the parameters as:

$$\mathrm{p}(b|S,\theta) = \prod_{t=1}^{T} \mathrm{p}(b(t)|S(t),\theta) = \prod_{t=1}^{T} \mathcal{N}(b(t); AS(t) + \mu_n, \Sigma_n) \tag{2.11}$$

where it is assumed that the factors $S(t)$ and noise $n(t)$ are independent at different time. To estimate the parameters, one can apply point estimate methods on (2.11), for example a maximum-likelihood approach. However, the maximum-likelihood method is suitable when there is no latent variable (unobserved factors)

in the model. To overcome the latter issue, Expectation Maximizations (EM) algorithm suggests the following iterates to find the appropriate value of the parameter θ:
Expectation step: Find the posterior of S:

$$\mathrm{p}(S|b, \theta^{t-1}) \tag{2.12}$$

Maximization step: Find the parameter that maximizes the following:

$$\theta^t = \arg\max_{\theta} E_{\mathrm{p}(S|b,\theta^{t-1})}[\log\ \mathrm{p}(S, b|\theta)] \tag{2.13}$$

where $E_q[.]$ denotes the expectation over q. Expectation maximization is a method for maximizing the likelihood of the parameters where there is some unobserved latent variables in the model. EM guarantees that the likelihood will not decrease at each iteration. After estimating the parameters, θ, one can find the most probable values of the factors S via the Maximum a Posteriori (MAP) method as:

$$S_{MAP} = \arg\max_{S} \mathrm{p}(S|b, \theta^o) \tag{2.14}$$

where θ^o is evaluated after the convergence of the EM algorithm.
Although by making use of factor analysis one can easily represent a set of observed data in terms of a set of uncorrelated factors but its basic version can not be used to represent the observed data in terms of a set of independent components (sources) [175]. In addition, the prior Gaussianity assumption about the factors reduce the flexibility of the FA in learning the linear models in which the sources (factors) have distributions other than Gaussian.

2.5 Independent Component analysis

Independent component analysis [37,39,94] is a statistical approach that aims to estimate a set of individual components from their mixture through a transformation over the mixed data as:

$$y = Wb \tag{2.15}$$

where W is the transformation matrix and y is an estimate of the individual components.
Independency assumption over the individual components is the key for separation in the ICA based algorithms. There are different measurement tools that

can measure the independency, which we discuss in the following sections.

2.5.1 Maximization of non-gaussianity

According to the knowledge extracted from the central limit theorem [13, 156], the distribution of a scalar random variable b_j defined as

$$b_j = \sum_{i=1}^{N} a_i S_i \tag{2.16}$$

where a_i's are sufficiently big constants and S_i's are independent identically distributed scalar random variables, tends to have a Gaussian distribution regardless of the distribution of S_i's. So we can see the connection between the dependency of the variable b_j and its Gaussianity. The more the non-Gaussian a variable is, the more independent it is assumed, since it can not be represented as a weighted sum of a set of independent random variables [85]. Therefore one may need to find an appropriate transformation, W, over the observed data, as samples of the random variable b, that maximize the non-Gaussianity of the estimated components, y.

Kurtosis

One of the objective functions that can be used to measure the Gaussianity of a distribution is the Kurtosis [17, 41, 163]. Kurtosis is a measurement tool that measures the tailedness and describe the shape of a distribution. For a zero mean random variable x the kurtosis is defined as:

$$\text{kurt}(x) = E[x^4] - 3(E[x^2])^2 \tag{2.17}$$

where $E[.]$ denotes the expectation operator.
Kurtosis can be both positive and negative. For super-Gaussian distributions, distributions that decay at least as fast as Gaussian, kurtosis is positive while for the sub-Gaussian distributions, those that decay at most as fast as Gaussian, kurtosis is negative. Interestingly for a Gaussian distribution the kurtosis is zero and as was mentioned earlier it is a non-zero value for most non-Gaussian distributions. Therefore maximizing the absolute value of the kurtosis is an appropriate objective for maximizing the non-gaussianity of the data. However,

the kurtosis is so sensitive to the outliers [64,82,121]. Although kurtosis has been used for ICA [106,114] it is not considered as a robust method for measuring the non-gaussianity since it is evaluated from the measured data, which may contain irrelevant samples.

Neg-entropy

Another effective objective that can measure the Gaussianity is the neg-entropy [26, 67]. In fact it is an optimal estimator of the Gaussianity [87]. For a scalar random variable x the neg-entropy is defined as:

$$J(x) = H(x_{gauss}) - H(x) \tag{2.18}$$

where x_{gauss} is a Gaussian random variable with the same variance as x and $H(x)$ is the differential entropy [40, 144] defined as:

$$H(x) = -\int \mathrm{p}(x)\mathrm{logp}(x)\mathrm{d}x \tag{2.19}$$

where $\mathrm{p}(x)$ is the probability density function of x.
According to the information theory, the differential entropy of a random variable is a measure of information content of the variable. Among all the distributions with the same variances, Gaussian distribution has the maximum deferential entropy. Therefore one can use (2.18) as a distance to normality. Neg-entropy is a non-negative value and is only zero for Gaussian distributions. So maximization of the neg-entropy can lead to estimation of the independent components. However, using (2.18) requires the estimation of the probability density functions, p(.), from the observed data, which is computationally demanding. The latter issue caused to use an approximate estimate of the neg-entropy. An example approximation of the neg-entropy for a zero mean unit variance random variable x is proposed in [83] as:

$$J(x) \propto [E[G(x_{gauss})] - E[G(x)]]^2 \tag{2.20}$$

where G can be any non-quadratic function. The most useful choices of the function G includes:

$$G_1(x) = \frac{1}{a1}\mathrm{logcosh}(a_1 x), \quad G_2(x) = -\exp(-\frac{x^2}{2}), \quad G_3(x) = \frac{x^4}{4} \tag{2.21}$$

where $1 \leq a_1 \leq 2$ is a constant. It is shown in [84] that G_2 and G_3 are more appropriate when the underlying independent components have super-Gaussian and sub-Gaussian distributions respectively while G_1 is a general purpose function.

2.5.2 Minimization of average mutual information

Another approach for measuring the independency inspired from the information theory is using the concept of average mutual information. Mutual information between two random variables y_1 and y_2, quantifies the similarity between their joint distribution, $\mathrm{p}(y_1, y_2)$, and the product of their marginal distribution, $\mathrm{p}(y_1)\mathrm{p}(y_2)$. For a set of independent random variables y_i, $i = 1, ..., N$ the average mutual information is zero and the following equality holds:

$$\mathrm{p}(y_1, ..., y_N) = \prod_{i=1}^{N} \mathrm{p}(y_i) \tag{2.22}$$

Average mutual information between a set of random variables y_i, $i = 1, ..., N$ (not necessarily independent), measures the distance between the left and right side of the equality in 2.22 via the following formula:

$$I(y_1, ..., y_N) = \sum_{i=1}^{N} H(y_i) - H(y_1, ..., y_N) \tag{2.23}$$

where H denotes the differential entropy.

According to [144] the average mutual information between a set of random variables y_i that are obtained via an invertible linear transformation $y = Wb$, where $y = [y_1, ..., y_N]^T$, can be written as:

$$I(y_1, ..., y_N) = \sum_{i=1}^{N} H(y_i) - H(b) - \log|\det W| \tag{2.24}$$

By comparing 2.24 with 2.18 and assuming that the variables y_i, $i = 1, ..., N$ in (2.24) have equal variances, we establish the following equality:

$$I(y_1, ..., y_N) = C - \sum_{i=1}^{N} J(y_i) \tag{2.25}$$

where C is a constant.

(2.25) shows the relationship between the neg-entropy and average mutual information. From (2.25) we see that minimization of the average mutual information of a set of random variables is equivalent to maximization of the non-gaussianity of each individual variable. Therefore by minimizing the average mutual information between the transformed version of the observed data one can estimate the independent components. The separation can be achieved by finding the appropriate transformation matrix W that minimizes $I(y_1 = W_1 b, ..., y_N = W_N b)$, where W_i denotes the i^{th} row of the matrix W. By making use of numerical optimization techniques such as gradient descent one can evaluate the appropriate de-mixing matrix W.

2.5.3 Maximization of information

As was mentioned earlier, the entropy of a random variable measures the information content of the variable. From the coding theory, we know that the join entropy of a set of random variables y_i, $i = 1, ..., N$ is the coding length of the variables. In fact it measures the required coding length for coding a set of random variables in the unit of bits. The more independent the variables are, the more bits is required for coding. This results in using the joint entropy as an alternative measurement tool for measuring independency.

Maximizing the joint entropy of a set of random variables is equivalent to maximizing their independency. [20, 135] used the latter concept to estimate the independent components via the neural networks. To estimate the independent components via the neural networks one can consider the observed data, b, as the input to a network which its outputs is obtained as $\Phi_i(W_i b)$, where W_i is the i^{th} row of the matrix W and is the weight vector of the i^{th} neuron and $\Phi_i(.)$ is a non-linear activation function, and find the optimal weight vectors that maximizes:

$$H(\Phi_1(W_1 b), ..., \Phi_N(W_N b)) \tag{2.26}$$

with suitable choice of non-linear function, $\Phi_i(.)$, maximization of (2.26) leads to separation of the independent components [20]. In fact, the estimated sources are calculated by applying learning rules on the neural networks whose activation functions are able to capture the underlying distribution of the original source signals.

2.5.4 Maximization of likelihood function

Making use of the generative model in (2.3), one can estimate the independent components, S_i, by maximizing the likelihood of the model. The likelihood function for a set of observed data $b(t)$, $t = 1, ..., T$ given the model (2.3) is written as:

$$\mathrm{p}(b|A) = \prod_{t=1}^{T} \mathrm{p}(b(t)|A) \tag{2.27}$$

where $\mathrm{p}(b(t)|A)$, the probability of the data at time index t is evaluated as:

$$\mathrm{p}(b(t)|A) = \int \mathrm{p}(b(t)|A, S)\mathrm{p}(S)\mathrm{d}S \tag{2.28}$$

Including the prior independency assumption about the random variables S_i, $i = 1, ..., N$ as $\mathrm{p}(S) = \prod_{i=1}^{N} \mathrm{p}(S_i)$ into the equation (2.28) and maximizing (2.27) leads to separation of the independent components [109]. For numerical stability, maximization is usually performed over the log of (2.27). Optimizing (2.27) leads to an optimization in terms of mixing matrix A. Alternatively, one can formulate a likelihood function in terms of the de-mixing matrix W. According to the probability theory, the probability of $b = AS$ can be written in terms of the probability of S and the linear transformation matrix A as:

$$\mathrm{p}_b(b) = \frac{1}{|\mathrm{det}A|}\mathrm{p}_S(S) \tag{2.29}$$

by substituting $W = A^{-1}$ and making use of the model in (2.3), (2.29) can be written as:

$$\mathrm{p}_b(b) = |\mathrm{det}W|\mathrm{p}_S(Wb) \tag{2.30}$$

Including the prior independency assumption about the variables S_i, $i = 1, ..., N$, for a set of observed data across time $t = 1, ...T$ the likelihood can be formulated as:

$$L(W) = \prod_{t=1}^{T} \mathrm{p}_b(b(t)) = |\mathrm{det}W|^T \prod_{t=1}^{T}\prod_{i=1}^{N} \mathrm{p}_i(W_i b(t)) \tag{2.31}$$

for numerical stability purposes, one can use the log of (2.31) as the objective function:

$$\mathrm{log}L(W) = \sum_{t=1}^{T}\sum_{i=1}^{N} \mathrm{log}\mathrm{p}_i(W_i b(t)) + T\mathrm{log}|\mathrm{det}W| \tag{2.32}$$

By comparing (2.32) with (2.24) we see that the following relationship holds:

$$\text{log}L(W) = -TI(y_1, ..., y_N) - TH(b) \tag{2.33}$$

where $H(b)$ is a constant. Therefore, maximization of log-likelihood, $\text{log}L(W)$, is equivalent to minimization of the average mutual information.

2.5.5 Natural gradient based optimization

After obtaining a variety of objective functions for independent component analysis, one needs to apply numerical optimization methods over those functions to extract the independent components. The standard stochastic gradient [7, 189] seems to be one solution for this purpose. However, applying the standard stochastic gradient is only meaningful over parameter spaces with Euclidean structure.

It has been observed that the parameter space of blind source separation, the space of non-singular matrices, behaves like a Riemannian manifold. In a non-linear space like the space of non-singular matrices, the standard gradient descent does not give the steepest direction of the objective function but the natural gradient [8] does. The natural gradient of a function $K(W)$ is defined as:

$$\tilde{\nabla}K(W) = Z^{-1}\nabla K(W) \tag{2.34}$$

where ∇ denotes the ordinary gradient and the matrix Z is the Riemannian metric tensor.

It has been demonstrated in [8] that the space of invertible matrices W has a metric tensor as $Z = (WW^T)^{-1}$. Using the invariance property of the inner product under translation in a Riemannian space, [14, 174] introduced different natural gradient formulations for the parameter space of blind source separation as:

$$\begin{aligned}
\tilde{\nabla}_L K(W) &= WW^T\nabla K(W) \\
\tilde{\nabla}_R K(W) &= \nabla K(W)W^T W \\
\tilde{\nabla}_{LR} K(W) &= WW^T\nabla K(W)W^T W \\
\tilde{\nabla}_{RR} K(W) &= \nabla K(W)W^T W^T WW \\
\tilde{\nabla}_{LL} K(W) &= WWW^T W^T\nabla K(W)
\end{aligned} \tag{2.35}$$

Considering the average mutual information as the objective function and applying the ordinary gradient leads to the following update [11]:

$$W(t+1) = W(t) + \mu(I - g(y^T(t))y(t))W^{T^{-1}}(t) \tag{2.36}$$

where $g(.)$ is a non-linear function and μ is the learning rate.
Applying the natural gradient in the form of $\tilde{\nabla}_R$ instead of the ordinary gradient leads to a more numerical stable update as:

$$W(t+1) = W(t) + \mu(I - g(y^T(t))y(t))W(t) \tag{2.37}$$

2.37 is more numerically stable than 2.36 since it does not include the inverse of the matrix W.

2.6 Difficulties and ambiguities

One of the ambiguities that exists in independent component analysis is about the correct scaling of the estimated sources. As there is neither information about the mixing matrix A nor about the sources S any scaled version of the sources as RS, where R is an arbitrary diagonal scaling matrix, can still satisfy the ICA model as:

$$b = AR^{-1}RS \tag{2.38}$$

Another ambiguity that exist in ICA is about the correct permutation of the estimated sources. In fact any permuted version of the sources as PS, where P is the permutation matrix, can also be a solution given by the ICA algorithms as it also satisfies the model:

$$b = AP^{-1}PS \tag{2.39}$$

Therefore the following relationship holds between the estimated sources, y, as the output of the ICA algorithms and the original sources, S:

$$y = RPS \tag{2.40}$$

However, learning a scaled and permuted version of the sources is not too problematic in many applications especially in the cocktail-party problem in a delay-restricted environment using time domain approaches where the underlying sources are speech or music signals.

In a delay-restricted environment, the instantaneous linear model 2.3 is an appropriate generative model while in an echoic environment the following model mimics the real nature of the mixing process between the j^{th} source and i^{th} sensor (microphone):

$$b_i(t) = h_{ij}(t) * S_j(t) \tag{2.41}$$

where $h_{ij}(t)$ is the impulse response of the room between the j^{th} source and i^{th} microphone and $*$ denotes the convolution operator.

To be able to apply the linear multiplicative ICA algorithms over the convolutive models with low computational complexity, one can divide the observed data into a set of frames and then apply the short-time Discrete Fourier Transform (DFT) on each frame to produce a linear multiplicative model for each frequency bin as:

$$b(f,k) = H(f)S(f,k) \tag{2.42}$$

where $H(f)$ is a complex mixing matrix in the frequency domain and k denotes the frame index.

The multiplicative ICA algorithms can easily be applied on (2.42) for a set of frames, $k = 1, ..., K$, to estimate $S(f,k)$. After estimating $S(f,k)$, we can reconstruct the time series signal $S(t)$ at the frame index k by using the inverse DFT. However, reconstructing the spectrum of the signal correctly requires to have a correct knowledge about the permutation and scaling of the estimated spectrum at each frequency bin. This is because the spectrum at the frame k is reconstructed by concatenating $S(f,k)$ across different frequency bins. Concatenation with different permutations do not allow to reconstruct the spectrum for each source signal correctly. The scaling ambiguity results in reconstructing a filtered spectrum of the signals. Therefore, permutation and scaling problems would be a burden in convolutive models. There are a variety of approaches that have addressed the aforementioned issue, which can be found in [53,88,165,178].

It is also good to mention that the multiplicative ICA algorithms have also the potential to be applied directly on, without the need for a transformation to the frequency domain, a set of observations that are recorded in an anechoic environment, where there is at least a delay between the source signals and their observed versions via the microphones, if there is a limit on the number of delays and sources. In fact, by considering the delayed versions of the sources as independent components the multiplicative ICA algorithms can be easily applied to estimate the sources and their delayed versions as long as the total number of sources and delayed versions do not exceed the number of observations (microphones). The latter condition guarantees that the ICA problem is not under-determined. The (over) determinacy is a necessary condition in the standard independent component analysis (ICA) algorithms as otherwise the decomposition is not guaranteed to be unique.

Regardless of the convolutive or instantaneous multiplicative model of the mixing process, to address the cocktail-party problem via BSS algorithms, there is the need of a set of sensors (microphones) to capture the mixture data. Blind source separation via wireless sensor networks has recently drawn more attention in comparison with the wired sensors because of low cost and broad areal coverage. As was mentioned in Chapter 1, in the context of wireless sensor networks research on BSS can be divided into centralized and de-centralized processing approaches. The centralized processing approach in general carries limitations such as powerful hardware requirements, high power consumption of the transmission and lack of scalability [63]. To overcome the latter issues we investigate distributed (in-network) processing solutions and its application in blind source separation in this thesis.

In the next chapter we discuss distributed algorithms for the unconstrained problems with separable objective function. We review the diffusion strategy and belief propagation on probabilistic graphical models as example techniques.

3 Distributed optimization techniques based on separability

Distributed and parallel processing is a fast developing research area. Various technologies are used to build distributed systems. A distributed system is a collection of autonomous agents (processors) referred to as nodes which are connected through a network. The connectivity of the nodes allows them to coordinate their activities and share the resources of the system. The cooperation between the nodes allows the user to perceive the system as a single computing unit.

The distributed computing has several advantages compared to the centralized computing, such as: lower cost, reduced processing time and higher flexibility [162]. Instead of investing on expensive mainframe and supercomputers for centralized computing, one can distribute the data processing over a few minicomputers that cost much less than mainframe machines. By splitting the problems into modules and allocating them to different agents that run simultaneously and in parallel, we can also reduce the processing time. Parallel processing is a branch of distributed processing in which all the machines simultaneously execute the same task in order to increase the processing speed. The other advantage of distributed computing is flexibility, meaning that individual nodes can be located at different locations, e.g. near the resources, to do a single global task. The aforementioned benefits motivate us to follow approaches towards data splitting, to be able to allocate them to different machines, for solving our big problems. On the other hand there are many network-structured optimization problems that arises in a variety of applications in engineering [22,111,125,192]. In those applications the problem inherently is separable since the objective function of the optimization forms a sum of partial functions. The separability of the objective function allows to develop distributed algorithms over a network of nodes.

The problems with separable objective function can be divided into two categories. The first category includes the unconstrained problems and the second one includes the constrained problems. In this chapter we discuss two example distributed methods, known as diffusion and Belief Propagation (BP) on Probabilistic Graphical Models (PGMs), of the unconstrained problems. In Chapter 4 and 5, we discuss the problems of the second category. We first start with a discussion on diffusion method in Section 3.1 followed by a discussion on PGMs in Section 3.2.

3.1 Diffusion

Consider the problem of minimizing a global cost function $J(w)$ which is the sum of N local cost functions $J_k(w)$, $k = 1, ..., N$ as:

$$\min_{w} J(w) = \sum_{k=1}^{N} J_k(w) \tag{3.1}$$

The problems of the form above arise in many applications such as target localization, online machine learning and distributed sensing. In the distributed sensing application one can consider a network of N agents that are interested in estimating the parameters of a physical model. In the latter scenario we can assume that each agent k receives a filtered version of a sequence, $u_k(i)$, in presence of additive noise as $d_k(i)$ over time $i \geq 0$. The relationship between $u_k(i)$

and $d_k(i)$ can be modelled as follows:

$$d_k(i) = \sum_{m=0}^{M-1} \beta_m u_k(i-m) + v_k(i), \quad i \geq 0 \tag{3.2}$$

where $\{\beta_0, ..., \beta_{M-1}\}$ are the parameters of the model and $v_k(i)$ is the additive noise.

The equation (3.2) can also be written in a vector form as:

$$d_k(i) = u_{k,i} w^o + v_k(i), k = 1, \ldots, N \tag{3.3}$$

where $w^o = [\beta_0, ..., \beta_{M-1}]^T$ and $u_{k,i} = [u_k(i), ..., u_k(i-M+1)]$.

Making use of the equation (3.3), to estimate the parameter vector w^o, [167] introduced a network objective as:

$$\min_w \sum_{k=1}^{N} E|d_k(i) - u_{k,i} w|^2 \tag{3.4}$$

where $\sum_{k=1}^{N} E|d_k(i) - u_{k,i} w|^2$ is the global cost function and E denotes the expectation operator.

Making use of the separability of the global cost function in (3.4), one can assign local cost function as $J_k(w) = E|d_k(i) - u_{k,i} w|^2$ to each agent k, $k = 1, ..., N$. To estimate the parameter w^o and satisfying the network objective, each agent k is then responsible in minimizing its own cost function $J_k(w)$ using its own resources, $u_k(i)$ and $d_k(i)$. There are many adaptive algorithms [68, 70], such as Least Mean Square (LMS), Normalized Least Mean Square (NLMS) and Recursive Least Square (RLS), that can be used for this purpose. For the purpose of simplicity [167] uses the LMS algorithm for minimizing $J_k(w)$ which leads to the following updates:

$$w_{k,i} = w_{k,i-1} + \mu u^*_{k,i}(d_k(i) - u_{k,i} w_{k,i-1}) \tag{3.5}$$

where μ is the step size and $u^*_{k,i}$ denotes the complex conjugate of $u_{k,i}$.

Assume a non-cooperative scenario in which each agent is interested in minimizing its local cost function via the LMS algorithm (3.5) without any cooperation with other agents. For this scenario the Mean Square Deviation of each agent,

$\mathrm{MSD}_{\mathrm{ncop},k}$, and the network, $\mathrm{MSD}_{\mathrm{ncop,network}}$, is evaluated as:

$$\mathrm{MSD}_{\mathrm{ncop},k} = \lim_{i\to\infty} E \parallel \widetilde{w}_{k,i} \parallel^2 \approx \frac{\mu M}{2}\sigma^2_{v,k} \tag{3.6}$$

$$\mathrm{MSD}_{\mathrm{ncop,network}} = \frac{\mu M}{2}\frac{1}{N}\sum_{k=1}^{N}\sigma^2_{v,k} \tag{3.7}$$

where $\widetilde{w}_{k,i} = w^o - w_{k,i}$ and $\sigma^2_{v,k}$ is the variance of the noise at agent k. Equations (3.6)-(3.7) allow to measure the performance of the network and each agent in correct estimation of w^o in a non-cooperative manner. For a detailed derivation of (3.6) and (3.7) the reader is referred to [166, p. 362].

It is good to compare the distributed non-cooperative scenario with the centralized processing scenario. By making use of the LMS algorithm, in a centralized processing manner the parameter w is updated as follows:

$$w_i = w_{i-1} + \mu\frac{1}{N}\sum_{k=1}^{N} u^*_{k,i}(d_k(i) - u_{k,i}w_{i-1}) \tag{3.8}$$

It has been shown in [167] that using (3.8) leads to a MSD as:

$$\mathrm{MSD}_{\mathrm{cent}} = \frac{\mu M}{2}\frac{1}{N}\left(\frac{1}{N}\sum_{k=1}^{N}\sigma^2_{v,k}\right) \tag{3.9}$$

Comparing (3.9) with (3.7) we see a decrease of performance by a factor of $\frac{1}{N}$ in the non-cooperative strategy compared to the centralized processing. Therefore we can conclude that splitting a global objective and allocating modules to a set of agents without considering additional assumptions may lead to a lower performance compared to the centralized approach. A natural solution for maintaining the performance when applying the distributed processing is to build a communication platform between the nodes of the network. Diffusion strategy [167] uses the latter solution and propose two distributed algorithms known as Combine then Adapt (CTA) and Adapt then Combine (ATC).

Fig. 3.1 shows a sample communication graph of diffusion strategy. As it can be seen from the Fig. 3.1 each edge that connects two neighbour agents is weighted by two scalars. For convergence purposes the scalars should satisfy the following

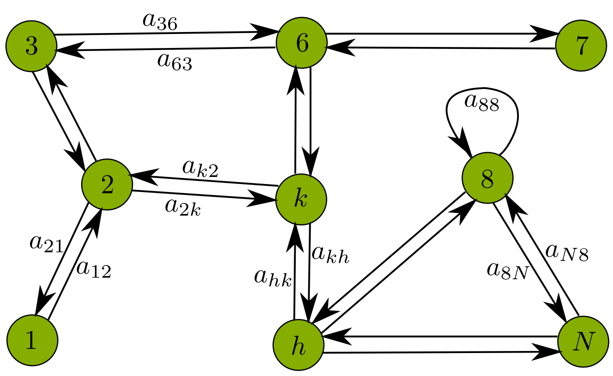


Figure 3.1: Illustration of a sample communication graph of diffusion strategy with N nodes. The neighbour of node k, denoted by $N(k)$, includes $\{2, h, 6, k\}$.

properties [167]:

$$a_{hk} \geq 0, \qquad \sum_{h \in N(k)} a_{hk} = 1, \qquad a_{hk} = 0 \text{ if } h \notin N(k) \tag{3.10}$$

where $N(k)$ denotes the neighbours of node k include k itself. In fact the matrix $A \equiv [a_{hk}]$, which is made by collecting the scalars a_{hk}, $h, k = 1, ..., N$, would be a left stochastic matrix as the result of consideration of the conditions in (3.10). A left stochastic matrix is a real square matrix, with each column summing to one. This property of the matrix A allows the agents to reach to a consensus about the estimation of the parameter vector w^o, since A satisfies the required conditions for development of distributed averaging algorithms over network of agents, see, e.g. [191].

Making use of the condition (3.10), the CTA algorithm uses the following updates in a cooperative manner to estimate the parameters of the model:

$$\begin{aligned} \psi_{k,i-1} &= \sum_{h \in N(k)} a_{hk} w_{h,i-1} \\ w_{k,i} &= \psi_{k,i-1} + \mu u^*_{k,i}(d_k(i) - u_{k,i}\psi_{k,i-1}) \end{aligned} \tag{3.11}$$

This means that each agent k is responsible in updating its expectation about the parameter w^o, denoted by w_k, using the expectation of the neighbour agents and then send it to its neighbour agents. Consequently the information that needs to be transmitted between two neighbour agents in the network is their expectation about the parameter vector.

By changing the order of the combination step and adaptation step, ATC algorithm suggests the following updates:

$$\begin{aligned}\psi_{k,i} &= w_{k,i-1} + \mu u^*_{k,i}(d_k(i) - u_{k,i}w_{k,i-1})\\ w_{k,i} &= \sum_{h\in N(k)} a_{hk}\psi_{k,i}\end{aligned} \tag{3.12}$$

It has been demonstrated in [207] that the diffusion methods (CTA & ATC) have a comparable performance to the centralized method. For a doubly stochastic matrix A (a real square matrix, with each column and row summing to one) the MSD of the diffusion methods is evaluated as [167]:

$$\mathrm{MSD}_{\text{diff},k} \approx \mathrm{MSD}_{\text{diff,network}} \approx \frac{\mu M}{2}\frac{1}{N}\left(\frac{1}{N}\sum_{k=1}^{N}\sigma^2_{v,k}\right) \tag{3.13}$$

Although requiring the matrix A to be a doubly stochastic matrix is more than the requirement that is needed for convergence, it allows the agents to have the same performance as the network. It is good to notice that in distributed sensing via diffusion strategy, each node of the network captures the same physical behaviour (medium) for different excitation signals. In diffusion strategy it is also assumed that the nodes have knowledge about the excitation signals and the only unknown is the parameter of the physical model. This is in contrast to the blind source separation scenario where both the parameters of the physical model and the source signals are unknown and each node receives the source signals through different physical transmission medium. Therefore, the diffusion algorithms (CTA and ATC) that were discussed in this section can not be directly applied for distributed BSS. However, by giving an explanation about the diffusion strategy and its application on distributed sensing, we saw that how a separable objective function can be allocated to a set of agents. We also noticed the importance of communication between the agents in distributed systems. In the next section we show how inference on probabilistic graphical models can lead to a distributed algorithm through an example application in solving the systems of linear equations.

3.2 Belief propagation on PGMs

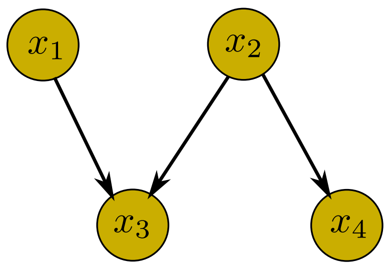
Graphical representation of probabilistic models of a set of random variables can help us to visualize the conditional dependence structure of the random variables [185]. Making use of the conditional dependence structure enables us to optimize the computational complexities of the inference problems. There are different types of probabilistic graphical models such as, Bayesian network, Markov network and Factor graphs [103].

In PGMs each vertex denotes a random variable and each edge express the dependency between the variables that are connected via that edge. Fig. 3.2 visualizes different possible graphical models for a set of random variables, (x_1, x_2, x_3, x_4), whose joint probability distribution is represented as

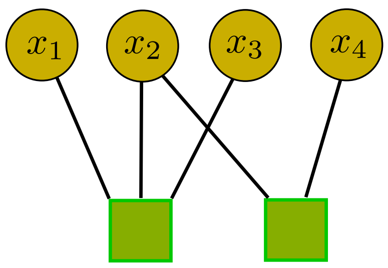
$$\mathrm{p}(x_1, x_2, x_3, x_4) = \mathrm{p}(x_4|x_2)\mathrm{p}(x_3|x_1)\mathrm{p}(x_3|x_2)\mathrm{p}(x_2)\mathrm{p}(x_1) \tag{3.14}$$

From the Fig. 3.2 we see that among the depicted graphical models, Markov graph reveals interesting properties compared to the Bayesian and Factor graphs. The properties include the non-discriminatory representation of the nodes compared to Factor graphs and non-causal (unidirectional) relationship between the nodes compared to the Bayesian graph. The discrimination between the nodes in Markov graph is eliminated by considering one set of corresponding nodes. This is in contrast to the factor graph that is a bipartite graph consisting of two different set of nodes, where each set is comprised of corresponding nodes. The aforementioned properties make a connection between the Markov graph and the communications graph of distributed systems. Thus, creating any distributed algorithm over the Markov graph can be easily applied on the graph of distributed systems.

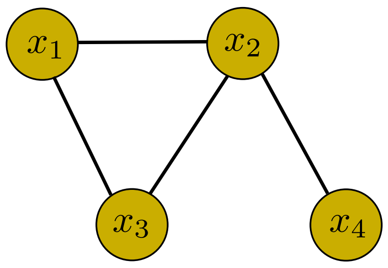
There are many inference methods on PGMs that lead to distributed algorithms. Examples of the inference methods are sum-product, max-product and max-sum [34]. The latter methods use the message passing (belief propagation) techniques [132] to calculate either the marginal distribution or the most probable state of the random variables. We now through an example inference on Markov graph show how a distributed algorithm for solving the system of linear equations can be built. Solving the system of linear equations has many applications in engineering and information science.



(a) Bayesian graph



(b) Factor graph



(c) Markov graph

Figure 3.2: Illustration of graphical models for a set of random variables whose joint probability distribution is represented as $\mathrm{p}(x_1, x_2, x_3, x_4) = \mathrm{p}(x_4|x_2)\mathrm{p}(x_3|x_1)\mathrm{p}(x_3|x_2)\mathrm{p}(x_2)\mathrm{p}(x_1)$.

Consider the problem of solving a system of linear equations:

$$b = Ay \tag{3.15}$$

where $b \in \mathbb{R}^N$ is the observation vector, $A \in \mathbb{R}^{N \times N}$ is a data matrix whose elements are the coefficients of the system and the objective is to find the unknown vector $y \in \mathbb{R}^N$. By assuming that the matrix A is full rank, the solution of the above problem is obtained via the following equation:

$$y = A^{-1}b \tag{3.16}$$

It has been demonstrated in [169] that for a symmetric matrix A the minimizer of the following quadratic function, $q(S)$, is the solution of the systems of linear equations:

$$q(S) = \frac{S^T A S}{2} - b^T S \tag{3.17}$$

Proof (for a symmetric matrix A):

$$\frac{\mathrm{d}q(S)}{\mathrm{d}S} = AS - b \tag{3.18}$$

setting (3.18) to zero results in:

$$AS = b \quad \Rightarrow \quad S = A^{-1}b \tag{3.19}$$

Therefore any function, $f(S)$, of the form below would have its maximizer at $S = A^{-1}b$:

$$f(S) = \frac{1}{z}\exp(-q(S)) \tag{3.20}$$

where z is a positive constant. Denoting $A^{-1}b$ by μ and by expanding (3.20), [169] established the following equality:

$$\begin{aligned} f(S) &= \frac{1}{z}\exp(-\frac{S^TAS}{2} + b^TS) \\ &= \frac{1}{z}\exp(\frac{\mu^TA\mu}{2})\exp(-\frac{S^TAS}{2} + \mu^TA\mu - \frac{\mu^TA\mu}{2}) \\ &= \frac{1}{\zeta}\exp(-\frac{(S-\mu)^TA(S-\mu)}{2}) \\ &= \mathcal{N}(\mu, A^{-1}) \end{aligned} \tag{3.21}$$

where $\zeta = z\exp(-\frac{\mu^TA\mu}{2})$ and $\mathcal{N}(\mu, A^{-1})$ denotes a Gaussian distribution with mean vector μ and covariance matrix A^{-1}.

According to the equality in (3.21), finding the maximizer of the function $f(S)$ corresponds to finding the most probable state of the Gaussian distribution $\mathcal{N}(\mu, A^{-1})$.

Thus, the problem of finding the solution of the system of linear equation can be formulated as:

$$y = \arg\max_S f(S) \tag{3.22}$$

where $f(S) = \mathcal{N}(\mu, A^{-1})$ is a multi-variable Gaussian distribution.

Because of the factorability property of the Gaussian distribution, we can split the objective function $f(S)$ as below:

$$f(S) \propto \prod_{i=1}^{N} \Phi_i(S_i) \prod_{\{i,j\}} \Psi_{ij}(S_i, S_j) \tag{3.23}$$

where $\Phi_i(S_i) = \exp(b_iS_i - \frac{A_{ii}S_i^2}{2})$, $\Psi_{ij}(S_i, S_j) = \exp(-S_iA_{ij}S_j)$ and S_i denotes the i^{th} element of the vector S. To solve the problem (3.22), we can apply the inference methods on a Markov graph that is induced from the separable Gaussian distribution (3.23), where the functions $\Phi_i(S_i)$ and $\Psi_{ij}(S_i, S_j)$, $\forall i, j =$

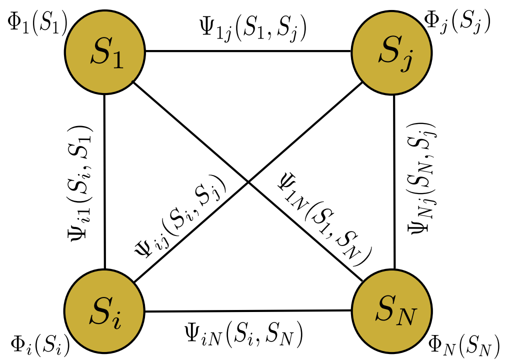


Figure 3.3: Illustration of a Markov graph that corresponds to a separable Gaussian distribution p(S), where p$(S) \propto \prod_{i=1}^{N} \Phi_i(S_i) \prod_{\{i,j\}} \Psi_{ij}(S_i, S_j)$. For the purpose of simplicity only 4 variables, $(1, i, j, N)$ are depicted.

$1, ..., N$, are appropriately allocated to the nodes of the graph. Fig. 3.3 illustrates the Markov graph that is induced from the distribution in (3.23), where A is a symmetric dense matrix, and shows how the functions $\Phi_i(.)$ and $\Psi_{ij}(., .)$ are allocated to the nodes and edges. It is good to note that the connectivity of the nodes of the graph in Fig. 3.3 depends on the non-zero elements of the matrix A. This is because the conditional dependency of the variables S_i and S_j depends on the ij^{th} element of the matrix A.

Allocation of the partial functions $\Phi_i(S_i)$ and $\Psi_{ij}(S_i, S_j)$, $\forall i, j = 1, ..., N$, into different nodes converts the inference problem (3.22) into a distributed inference problem. It has been shown in [169] that applying the max-product as inference tool for maximizing $f(S)$ on the Markov graph, leads to an iterative procedure in which each vertex i will be responsible for estimating its own variable S_i through cooperation with its neighbouring nodes. The messages that need to be transmitted between the nodes will be in the form of scalar Gaussian distributions that are usually encoded with two scalars, mean and variance. Because of the Gaussianity of the messages the method is also referred to as Gaussian Belief Propagation (GaBP).

By applying the GaBP algorithm [169] on a distributed system whose communication graph corresponds to a Markov graph that is induced from the distribution $\mathcal{N}(\mu, A^{-1})$, we are able to find the solution of the system of linear equations distributively and cooperatively, in which each agent is responsible for estimating its own unknown variable.

Distributed processing of the system of linear equations via the probabilistic approach not only leads to an iterative algorithm [15] that is inherently simple, requiring only additions and multiplications, but also allows us to exploit the *sparsity* structure of the matrix A and therefore reduce the computational complexity of calculating the direct matrix inversion in large scale problems [164].

It is important to note that the convergence of the GaBP, as a solver of system of linear equations, is restricted to symmetric and walk-summable matrices. A positive definite matrix A, with all ones on its diagonal, is walk-summable if the spectral radius of the matrix $\bar{B} = [|B_{ij}|]_{i,j=1}^{n}$, where $B = I - A$, is less than one. However, the convergence of GaBP for general matrices was fixed in [92] with the cost of adding additional iterations and communications between the nodes.

There are other variants of GaBP algorithm such as Linear Coordinate Descent (LiCD) [201] and Generalized Linear Coordinate Descent (GLiCD) [200] message passing algorithms that are optimized in terms of computation, storage and required transmission bandwidth between the nodes. An example application of GLiCD in distributed beamforming can be found in [77].

Distributed algorithms for solving the system of linear equations, for example GaBP or GLiCD, can also be applied in distributed blind source separation. For example in ICA algorithms that are based on estimating the mixing matrix, after evaluating the appropriate mixing matrix A, one can estimate the sources S as $S = A^{-1}b$, where b denotes the observed data. By using the GLiCD algorithm we can estimate the sources in a distributed fashion over a network of nodes, where each node i has access to his own observed data b_i and is responsible in estimating its own source S_i, given an estimation on the mixing matrix. However, the convergence of GaBP and GLiCD for general matrices A, (not necessarily symmetric walk-summable), requires considerable amount of communication between the nodes. In wireless sensor networks, where the energy of each sensor node is supplied by a small battery, high communication will decrease the life-time of the sensor nodes. Therefore we do not pursue applying GLiCD as a distributed algorithm for source estimation in this thesis.

In the next chapter we consider a globally constrained optimization problem and investigate distributed processing algorithms as solutions for the problem. There are a variety of problems in signal processing and machine learning which can be cast into the globally constrained optimization problem, for example robust principle component analysis [28, 208], basis pursuit problem [27, 33] and image denoising [153, 160].

4

A communication-efficient asynchronous distributed algorithm

4.1 Introduction

In this chapter, we consider linearly constrained convex optimization problems with $N \geq 2$ blocks of variables where the constraint includes a weighted sum of all the decision variables. These forms of problems arise in many applications such as machine learning, compressive sensing and statistics [28,31,58,131,145,148,176]. For problems with separable objective function, it is desirable to develop parallel algorithms to solve the problem. Parallelization of the algorithm facilitates the optimization of the objective function for the case of a very large number of decision variables.

To address the constrained problem, it is first converted to an unconstrained optimization by forming a Lagrangian function [21,181]. Assuming that a solution exists for the constrained problem, the optimal point of the decision variables together with the Lagrange multiplier (dual variable) will be evaluated at the saddle point of the Lagrangian function [24,181]. To evaluate the optimal point, one can first formulate a dual problem, find the optimal point of the dual variable from the dual function, and then find the optimal point of the decision variables from the Lagrangian function [25]. To achieve the latter it is easier to start from an initial point and follow a primal-dual algorithm in which the dual and primal variables are evaluated from the Lagrangian function one after another in an iterative procedure.

An example primal-dual algorithm is the dual ascent method or dual sub-gradient method [25,170]. One of the challenges in the dual ascent method is that specific assumptions need to be satisfied for convergence. Given the convergence guarantee, the dual ascent method has a slow rate of convergence of $O(1/\sqrt{k})$ for general convex functions, where k denotes the iteration counter. Although the dual ascent method does not benefit from a good convergence property it naturally leads to a parallel algorithm known as dual decomposition [49] for problems with separable objective functions. Applying the dual decomposition algorithm over a network of agents suggests a star pattern of connectivity in which a master agent, to which all other worker agents are connected, is responsible for the update of the dual variable and the worker agents are responsible for the update of the primal or decision variables.

To improve the convergence property of the dual ascent method a penalty term is added to the Lagrangian function, forming an *augmented* Lagrangian [76]. A well-known primal-dual method that is applied over the augmented Lagrangian is the method of multipliers [76,127]. A drawback of the method of multipliers is that the primal variables need to be updated jointly, which no longer allows a parallel update. The latter deficiency is the consequence of the existence of the penalty term in the augmented Lagrangian, which couples all the primal variables.

Motivated by decoupling the primal update, the Alternating Direction Method of Multipliers (ADMM) [55] was proposed to add decomposability of the update of the decision variables to the method of multipliers. It was first introduced in 1970s and has been popular in recent years; a thorough review of the ADMM and its convergence analysis can be found in [24,48]. ADMM updates the primal variables alternately, similarly to a Gauss-Seidel procedure. It was developed for two blocks of variables ($N = 2$) and its convergence was investigated and proved in [54] for constrained problems with two blocks of decision variables. An extension of ADMM from two blocks of variables to $N > 2$ blocks of variables suggests a Gauss-Seidel iterative procedure over the updates of the primal variables and is known as Gauss-Seidel ADMM [72]. The problem with Gauss-Seidel ADMM is that the primal variables are updated sequentially, which does not allow for parallelization. It is more desirable to run the updates of the primal variables in Gauss-Seidel ADMM in parallel. An algorithm that satisfies the latter desire is Jacobi ADMM.

Although Gauss-Seidel ADMM and Jacobi ADMM have been applied successfully to some problems [187], neither of them is necessarily convergent for $N > 2$ blocks of variables [32]. [71] introduced and discussed specific assumptions by which the Jacobi ADMM is guaranteed to converge for multi-blocks of variables. Generally the requirement for the convergence of Jacobi ADMM consists of a conservative movement towards the saddle point of the augmented Lagrangian function [71, 75]. The penalty term in the augmented Lagrangian implies that a primal-dual algorithm based on the augmented Lagrangian over a network of agents imposes a fully connected network in which a master node is responsible for the update of the dual variable and all remaining (worker) agents are responsible for the update of the primal or decision variables.

In this chapter we develop a primal-dual algorithm that exploits both Lagrangian and augmented Lagrangian functions. We evaluate the primal variables from a Lagrangian type function to benefit from the full separability of the primal variables of the Lagrangian function. The difference between the proposed primal-dual algorithm and the dual decomposition algorithm is that the dual variable is evaluated from a dual function instead of being evaluated directly from the Lagrangian function. To be able to benefit from a parallel optimization in the dual problem, we convert the dual problem to a consensus problem in which the objective function would be separable across the agents. The advantage of converting the dual problem to a consensus problem for development of distributed algorithms for a specific kind of problems and its application to distributed signal processing has been recently highlighted in [?]. We address the consensus problem by formulating an augmented Lagrangian function and evaluate its saddle point using the Bi-ADMM algorithm [202] as a distributed algorithm. Making use of the aforementioned augmented Lagrangian, the proposed primal-dual algorithm converges both synchronously and asynchronously with the rate of $O(1/k)$ where k denotes the iteration counter for optimization of the consensus problem. The asynchronism of the updates of the variables allows the algorithm to be applied in real-world scenarios in which coordination of a set of distinct and disparate agents is demanding, for example in heterogeneous networks.

Another outcome of the proposed approach arises from converting the dual problem to a consensus problem which allows the primal-dual algorithm to be applied over any graph with any connectivity patterns as long as the graph is connected. By applying the proposed primal-dual algorithm over a sparse connected graph we can reduce the total communication complexity.

The remainder of this chapter is organized as follows; In Section 4.2 we review the related works and the algorithms that they introduce as a solution to the problem (4.1). In Section 4.3 we introduce the proposed algorithm. The evaluation of the proposed approach is presented in Section 4.4. It is followed by a conclusion in Section 4.5.

4.2 Related Work

Consider the following optimization problem with $N \geq 2$:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{N} A_i x_i = c \\ & x_i \in \chi_i \ , \quad i = 1, ..., N \end{aligned} \tag{4.1}$$

where $(x_i, A_i, c) \in (\mathbb{R}^{n_i}, \mathbb{R}^{m \times n_i}, \mathbb{R}^m)$, χ_i is a convex set and $f_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$ is a closed, proper and convex function.

In order to incorporate the constraint into the objective term in the problem (4.1) the Lagrangian function is constructed as:

$$L(x_1, ..., x_N, \lambda) = \sum_{i=1}^{N} f_i(x_i) - \lambda^T (\sum_{i=1}^{N} A_i x_i - c) \tag{4.2}$$

where $\lambda \in \mathbb{R}^m$ is the Lagrange multiplier or the dual variable.

Assuming that a solution exists for the problem (4.1) as $(x_1^*, ..., x_N^*)$, it can be

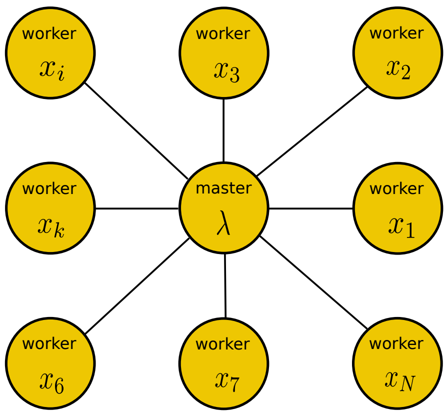


Figure 4.1: Illustration of the graph of dual decomposition method.

obtained by first evaluating the λ^* from the dual problem as:

$$\lambda^* = \arg\max_{\lambda} g(\lambda) \tag{4.3}$$

where $g(\lambda) = \inf_{x_1 \in \chi_1, \dots, x_N \in \chi_N} L(x_1, ..., x_N, \lambda)$ is the dual function and then calculating the $(x_1^*, ..., x_N^*)$ from the Lagrangian function as:

$$(x_1^*, ..., x_N^*) = \arg \min_{x_1 \in \chi_1, \dots, x_N \in \chi_N} L(x_1, ..., x_N, \lambda^*) \tag{4.4}$$

To simplify (4.3) and constructing a parallel optimization, the dual decomposition algorithm considers the following iterative updates as an alternative to (4.3)-(4.4) [49]:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i \in \chi_i} f_i(x_i) - \lambda^{k^T} A_i x_i \quad , i = 1, ..., N \\ \lambda^{k+1} &= \lambda^k - \alpha^k (\sum_{i=1}^{N} A_i x_i^{k+1} - c) \end{aligned} \tag{4.5}$$

where $\sum_{i=1}^{N} (A_i x_i - c)$ is the gradient of $g(\lambda)$ and k denotes the iteration counter. With a suitable choice of α^k and assuming that some specific assumptions hold [137] (e.g. $f_i(x_i)$ is not a nonzero affine function of x_i), the updates in (4.5) are guaranteed to converge to the saddle point, $(x_1^*, ..., x_N^*, \lambda^*)$, of $L(x_1, ..., x_N, \lambda)$. By following (4.5) the x_i-subproblems can be solved fully in parallel over a network of agents with a communication graph visualized in Fig. 4.1.

As was mentioned earlier the dual decomposition algorithm has a slow rate of convergence for general convex functions which makes it impractical in many applications. To improve the convergence properties of the dual decomposition method and bring robustness to it the method of multipliers was introduced [76,152]. The method of multipliers exploits the augmented Lagrangian function by adding a penalty term to the Lagrangian function as:

$$L_\rho(x_1, ..., x_N, \lambda) = \sum_{i=1}^{N} f_i(x_i) - \lambda^T (\sum_{i=1}^{N} A_i x_i - c) + \frac{\rho}{2} || \sum_{i=1}^{N} A_i x_i - c||_2^2 \qquad (4.6)$$

where ρ is a positive constant.

ADMM is the multipliers method that uses partial updates in an alternating manner similar to the Gauss-Seidel updates. ADMM was developed for two blocks of variables, $N = 2$, in which it updates the variables as follows:

$$\begin{aligned} x_1^{k+1} &= \arg \min_{x_1 \in \chi_1} L_\rho(x_1, x_2^k, \lambda^k) \\ x_2^{k+1} &= \arg \min_{x_2 \in \chi_2} L_\rho(x_1^{k+1}, x_2, \lambda^k) \\ \lambda^{k+1} &= \lambda^k - \rho(\sum_{i=1}^{N} A_i x_i^{k+1} - c) \end{aligned} \qquad (4.7)$$

ADMM converges in far more general conditions with a quicker rate of convergence than the dual decomposition method and its convergence for $N = 2$ has been proven [54]. A trivial solution to apply ADMM for multi-blocks of variables, $N \geq 2$, is to consider the following updates for x_i-subproblems:

$$x_i^{k+1} = \arg \min_{x_i \in \chi_i} L_\rho(x_1^{k+1}, ..., x_{i-1}^{k+1}, x_i, x_{i+1}^k, ..., x_N^k, \lambda^k) \qquad (4.8)$$

which leads to the well-known Gauss-Seidel ADMM algorithm.

Algorithm 1 Gauss-Seidel ADMM

```
Initialize $x_i$, $i = 1, ..., N$ and $\lambda$
while the stopping criteria is not met do
    for $i = 1, ..., N$ [sequentially] do
        $x_i^{k+1} = \arg\min_{x_i \in \chi_i} f_i(x_i) - \lambda^{k^T} A_i x_i +$
            $\frac{\rho}{2} || \sum_{j=1}^{i-1} A_j x_j^{k+1} + A_i x_i + \sum_{j=i+1}^{N} A_j x_j^k - c||_2^2$
    end for
    $\lambda^{k+1} = \lambda^k - \rho(\sum_{i=1}^{N} A_i x_i^{k+1} - c)$
    $k \leftarrow k + 1$
end while
```

Algorithm 1 shows the entire updating procedure in the Gauss-Seidel ADMM. As it can be seen from Algorithm 1 the primal variables are updated sequentially which no longer allows for parallelization. To be able to develop a parallel algorithm corresponding to Algorithm 1, one can replace the x_i updates with the following updates:

$$x_i^{k+1} = \arg \min_{x_i \in \chi_i} L_\rho(x_1^k, ..., x_{i-1}^k, x_i, x_{i+1}^k, ..., x_N^k, \lambda^k) \tag{4.9}$$

which leads to well-known Jacobi ADMM algorithm as shown in Algorithm 2. However, neither of the Jacobi and Gauss-Seidel versions of the ADMM are guaranteed to converge.

Algorithm 2 Jacobi ADMM

```
Initialize $x_i$, $i = 1, ..., N$ and $\lambda$
while the stopping criteria is not met do
    for $i = 1, ..., N$ [in parallel] do
        $x_i^{k+1} = \arg\min_{x_i \in \chi_i} f_i(x_i) - \lambda^{k^T} A_i x_i + \frac{\rho}{2} ||A_i x_i + \sum_{j \neq i} A_j x_j^k - c||_2^2$
    end for
    $\lambda^{k+1} = \lambda^k - \rho(\sum_{i=1}^{N} A_i x_i^{k+1} - c)$
    $k \leftarrow k + 1$
end while
```

[32, 42] showed that the mutual orthogonality of the matrices A_i is a sufficient condition for convergence of the Jacobi and Gauss-Seidel ADMM algorithms. In [65, 71, 75, 91] it was suggested to combine an under-relaxation step with the

output of the Jacobi ADMM to guarantee its convergence for general matrices A_i as:

$$w^{k+1} = w^k - \beta(w^k - \tilde{w}^k) \tag{4.10}$$

where $\beta > 0$ is a constant step size, $w^k = [x_1^k, ..., x_N^k, \lambda^k]^T$ is the input to the Jacobi ADMM algorithm and $\tilde{w}^k$ is its output.

Algorithm 3 UR-Jacobi ADMM

```
Initialize $x_i$, $i = 1, ..., N$ and $\lambda$
while the stopping criteria is not met do
    for $i = 1, ..., N$ [in parallel] do
        $\tilde{x}_i^{k+1} = \arg\min_{x_i \in \chi_i} f_i(x_i) - \lambda^{k^T} A_i x_i + \frac{\rho}{2}||A_i x_i + \sum_{j \neq i} A_j x_j^k - c||_2^2$
    end for
    $\tilde{\lambda}^{k+1} = \lambda^k - \rho(\sum_{i=1}^N A_i \tilde{x}_i^{k+1} - c)$
    Generate $w^{k+1}$ using (4.10) with $\beta = \eta(1 - \sqrt{\frac{N}{N+1}})$
    and $\eta \in (0, 2)$
    $k \leftarrow k + 1$
end while
```

Using (4.10) with an appropriate upper bound on β the Jacobi ADMM is guaranteed to converge. The difference between [65, 75, 91] and [71] is the assumptions that they consider for the value of N and β. In fact the algorithms in [65] and [75] are developed for $N = 2$ and $N = 3$ respectively while [91] and [71] are designed for $N \geq 2$. Between [91] and [71], [71] introduces a larger upper bound for β. The entire updating procedure of [71] is shown in Algorithm 3 and we refer to it as under-relaxation based Jacobi ADMM (UR-Jacobi ADMM). Another effective approach that has been proposed to guarantee the convergence of the Jacobi ADMM is the Proximal Jacobi ADMM [42]. In the Proximal Jacobi ADMM a proximal term of the form of $\frac{1}{2}||x_i - x_i^k||_{P_i}^2$, where P_i is a symmetric positive semi-definite matrix, is added to each x_i-subproblem and a damping parameter, $\gamma > 0$, controls the update of the dual variable as shown in Algorithm 4. The proximal terms together with the damping parameter play the same role as the under relaxation step (4.10) which leads to a conservative movement towards the saddle point of the augmented Lagrangian function. The advantage of the Proximal Jacobi ADMM compared to the UR-Jacobi ADMM is that the existence of the proximal terms can lead to an easier subproblem by choosing the appropriate

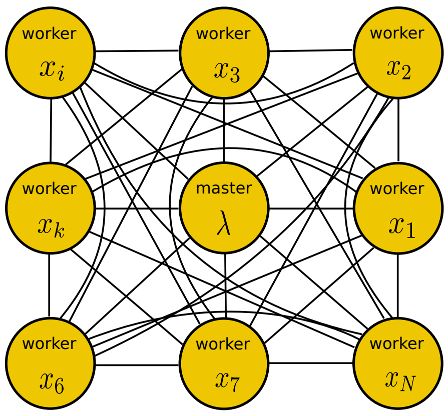


Figure 4.2: Illustration of the graph of methods based of augmented Lagrangian.

Algorithm 4 Proximal Jacobi ADMM

```
Initialize $x_i$, $i = 1, ..., N$ and $\lambda$
while the stopping criteria is not met do
    for $i = 1, ..., N$ [in parallel] do
        $x_i^{k+1} = \arg\min_{x_i \in \mathcal{X}_i} f_i(x_i) - \lambda^{k^T} A_i x_i + \frac{\rho}{2}||A_i x_i + \sum_{j \neq i} A_j x_j^k - c||_2^2 + \frac{1}{2}||x_i - x_i^k||_{P_i}^2$
    end for
    $\lambda^{k+1} = \lambda^k - \gamma\rho(\sum_{i=1}^N A_i x_i^{k+1} - c)$
    $k \leftarrow k + 1$
end while
```

matrices P_i. For example by choosing $P_i = D_i - \rho A_i^T A_i$, where D_i is an arbitrary well-conditioned diagonal matrix, one can cancels the quadratic term $\frac{\rho}{2} x_i A_i^T A_i x_i$ that appears in the x_i-subproblem. This cancellation would be an advantage when $A_i^T A_i$ is ill-conditioned or computationally expensive to invert. In addition the proximal terms make each subproblem strongly convex when they are just convex which leads to a more stable algorithm [42]. It is important to note that the aforementioned Algorithms 1, 2, 3 and 4 that utilize the augmented Lagrangian function require a communication graph as illustrated in Fig. 4.2. A master agent is responsible for the update of the dual variable and the worker agents are responsible for the update of the primal variables.

4.3 Proposed Approach

In this section we propose a primal-dual form algorithm that eliminates the need for a master agent. It utilizes a Lagrangian type function to evaluate the primal variables and an augmented Lagrangian type function to evaluate the dual variable from a constrained optimization problem. This results to enjoy the full parallelization of the x_i-subproblems of the Lagrangian function and the superior convergence property of the augmented Lagrangian based methods, e.g. a convergence rate of $O(1/k)$.

We split the dual variable by introducing N copies of it, denoted by $\lambda_1, ..., \lambda_N$. Then we consider the following Lagrangian function:

$$L(x_1, ..., x_N, \lambda_1, ..., \lambda_N) = \sum_{i=1}^{N} f_i(x_i) - \sum_{i=1}^{N} \lambda_i^T (A_i x_i - \frac{c}{N}) \tag{4.11}$$

The dual function of the above Lagrangian function can then be formulated as:

$$\begin{aligned} g(\lambda_1, ..., \lambda_N) &= \inf_{x_1 \in \chi_1, ..., x_N \in \chi_N} L(x_1, ..., x_N, \lambda_1, ..., \lambda_N) \\ &= \sum_{i=1}^{N} [\inf_{x_i \in \chi_i} \{f_i(x_i) - \lambda_i^T A_i x_i\} + \lambda_i^T \frac{c}{N}] \\ &= \sum_{i=1}^{N} [-f_i^*(A_i^T \lambda_i) + \lambda_i^T \frac{c}{N}] \end{aligned} \tag{4.12}$$

where f^* denotes the convex conjugate, also referred to as the Legendre Fenchel transform, of f [80,158]. By introducing a new local function $h_i(\lambda_i) = f_i^*(A_i^T \lambda_i) - \lambda_i^T \frac{c}{N}$ the dual problem associated to the Lagrangian function (4.11) can be written in the form of a consensus problem over a graph $G = (\nu, \varepsilon)$, where ν denotes the set of nodes with cardinality $N = |\nu|$ and ε denotes the set of edges, as:

$$\begin{aligned} &\min_{\lambda_i} \sum_{i \in \nu} h_i(\lambda_i) \\ &\text{s.t.} \quad \lambda_i = \lambda_j \quad \forall (i,j) \in \varepsilon \end{aligned} \tag{4.13}$$

The objective is now to find the optimal dual variable λ_i^*, $i \in \nu$ from (4.13) in a distributed manner over the graph G and to evaluate the optimal primal variables from the Lagrangian function (4.11) in parallel as:

$$x_i^* = \arg\min_{x_i \in \chi_i} f_i(x_i) - {\lambda_i^*}^T A_i x_i \tag{4.14}$$

The optimization problem consisting of (4.13) and (4.14) forms our proposed alternative optimization of problem (4.1). It is important to note that the proposed approach requires the calculation of the convex conjugate function compared to the traditional methods.

Numerous distributed algorithms can be used to solve consensus problems of the form of (4.13), both synchronously and asynchronously, e.g. [90, 188, 202, 206]. Among the latter algorithms the Bi-ADMM algorithm [202] activates one agent rather than one edge per iteration, thus eliminating the need for coordination between agents both locally and globally. Therefore, we use the Bi-ADMM algorithm to address (4.13) in this thesis. The Bi-ADMM algorithm exploits an augmented primal-dual Lagrangian function for problems of form (4.13) as:

$$\begin{aligned} L_I(\lambda_1, ..., \lambda_N, \delta) = & \sum_{i \in \nu} [h_i(\lambda_i) - \sum_{j \in N(i)\backslash i} \text{sign}(j-i)\delta_{j|i}^T \lambda_i - h_i^*(\sum_{j \in N(i)\backslash i} \text{sign}(j-i)\delta_{i|j})] \\ & \sum_{(i,j)\in\varepsilon} (\frac{1}{2}||\lambda_i - \lambda_j||_2^2 - \frac{1}{2}||\delta_{i|j} - \delta_{j|i}||_2^2) \end{aligned} \tag{4.15}$$

where h_i^* is the convex conjugate of h_i, $\delta = [\delta_1^T, ..., \delta_N^T]^T$, δ_i denotes the vector obtained by vertically concatenating all $\delta_{i|j}$, $j \in N(i)\backslash i$, $\delta_{i|j}$ is the dual variable associated to the constraint, $\lambda_i = \lambda_j$, on the edge (i, j) which is held at node i, and $N(i)$ denotes the neighbours of node i including i itself and $N(i)\backslash i$ excludes node i from its neighbours.

Using the above augmented primal-dual Lagrangian, the Bi-ADMM algorithm converges to its saddle point with $O(1/k)$ rate of convergence [204] both synchronously and asynchronously regardless of the topology of the graph G, as long as the graph is connected [202].

We consider the following two assumptions.

***Assumption* 1**: The KKT conditions [96, 97, 104] hold for the problem (4.1) as:

$$\begin{aligned} & A_i^T \lambda^* \in \partial f_i(x_i^*) \quad , i = 1, ..., N \\ & \sum_{i=1}^{N} A_i x_i^* = c \end{aligned} \tag{4.16}$$

In other words, there exists a saddle point $(x_1^*, ..., x_N^*, \lambda^*)$ for the Lagrangian function (4.2) that satisfies the following inequality:

$$L(x_1^*, ..., x_N^*, \lambda) \leq L(x_1^*, ..., x_N^*, \lambda^*) \leq L(x_1, ..., x_N, \lambda^*) \tag{4.17}$$

***Assumption* 2**: The augmented primal-dual Lagrangian function (4.15) has a saddle point.

Assuming that Assumption 2 holds, Bi-ADMM uses the following updates to iteratively approach the solution of (4.13):

$$\begin{aligned}\lambda_i^{k+1} =& \arg\min_{\lambda_i}[h_i(\lambda_i) + \sum_{j\in N(i)\backslash i}\{-\text{sign}(j-i)\lambda_i^T\delta_{j|i}^k + \frac{1}{2}||\lambda_i - \lambda_j^k||_2^2\}] \\ \delta_{i|j}^{k+1} =& \delta_{j|i}^k + \text{sign}(j-i)(\lambda_j^k - \lambda_i^{k+1})\end{aligned} \tag{4.18}$$

By combining (4.18) with (4.14) we can associate a primal point x_i^k to each dual point λ_i^k at each iteration, similarly to primal-dual algorithms. Algorithm 5 and 6 show the proposed distributed algorithm for the case of synchronous and asynchronous updates respectively. We refer to the proposed algorithms as dual decomposition Bi-ADMM (DDBi-ADMM).

Algorithm 5 DDBi-ADMM (synchronous version)

```
Randomly initialize $x_i, \lambda_i$ and $\delta_{i,j}$ $\forall i, j \in \nu$
do [in parallel]
    $x_i^{k+1} = \arg\min_{x_i\in\chi_i} f_i(x_i) - \lambda_i^{k^T} A_i x_i$
    $\lambda_i^{k+1} = \arg\min_{\lambda_i}[h_i(\lambda_i) + \sum_{j\in N(i)\backslash i}\{-\text{sign}(j-i)\lambda_i^T\delta_{j|i}^k + \frac{1}{2}||\lambda_i - \lambda_j^k||_2^2\}]$
    $\delta_{i|j}^{k+1} = \delta_{j|i}^k + \text{sign}(j-i)(\lambda_j^k - \lambda_i^{k+1}) \quad \forall\ j \in N(i)\backslash i$
    $k \leftarrow k+1$
while the stopping criteria is not met
```

Algorithm 6 DDBi-ADMM (asynchronous version)

```
Randomly initialize $x_i, \lambda_i$ and $\delta_{i,j}$ $\forall i, j \in \nu$
while the stopping criteria is not met do
    Randomly choose an agent $i$
    $x_i^{k+1} = \arg\min_{x_i \in \chi_i} f_i(x_i) - \lambda_i^{k^T} A_i x_i$
    $\lambda_i^{k+1} = \arg\min_{\lambda_i} [h_i(\lambda_i) + \sum_{j \in N(i) \backslash i} \{-\text{sign}(j-i)\lambda_i^T \delta_{j|i}^k + \frac{1}{2}||\lambda_i - \lambda_j^k||_2^2\}]$
    $\delta_{i|j}^{k+1} = \delta_{j|i}^k + \text{sign}(j-i)(\lambda_j^k - \lambda_i^{k+1}) \quad \forall\, j \in N(i) \backslash i$
    $(x_j^{k+1}, \lambda_j^{k+1}, \delta_j^{k+1}) = (x_j^k, \lambda_j^k, \delta_j^k) \quad \forall\, j \in \nu\ ,\, j \neq i$
    $k \leftarrow k + 1$
end while
```

From the Algorithm 5 and 6 we see that the synchronous version of the DDBi-ADMM algorithm is identical to the asynchronous version except that at each iteration all agents are active and update their variables x_i, λ_i and δ_i in parallel. One of the advantages of the proposed algorithm is that it can be implemented over any graph as long as the graph is connected, by connectivity we mean that the graph is not separated and there is at least a path between any pair of nodes. By applying the DDBi-ADMM algorithm over a sparse graph we can reduce the total number of communication processes per iteration. The communication process is referred to as data sharing between one pair of agents that are linked via an edge on the graph. For example by applying the proposed method over a sparse graph with $|\varepsilon|$ edges we can reduce the total communication processes by a factor $O(\frac{N(N-1)}{2|\varepsilon|})$ compared to the traditional algorithms that exploit the augmented Lagrangian function as the total number of edges over a dense graph is $\frac{N(N-1)}{2}$. It is important to note that the messages that are transmitted between two worker agents i and j in the augmented Lagrangian based methods are comprised of x_i and x_j. This is in contrast to the proposed approach in which the messages that are transmitted between two worker agents i and j are comprised of λ_i, λ_j, $\delta_{i|j}$, $\delta_{j|i}$. Since $(\lambda_i, \lambda_j, \delta_{i|j}, \delta_{j|i}) \in \mathbb{R}^m \; \forall\, (i,j) \in \varepsilon$, for a given communication channel on the edge (i,j) we can also save on the transmission bandwidth if $m < \frac{n_i+n_j}{4}$.

By comparing the Proximal Jacobi ADMM with the DDBi-ADMM algorithm we see that the convergence of the Proximal Jacobi ADMM is guaranteed only for $P_i = \tau_i I$ or $P_i = \tau_i I - \rho A_i^T A_i$ [42], where I is an identity matrix, with

$\tau_i > \rho(\frac{N}{2-\gamma} - 1)||A_i||^2$ or $\tau_i > \frac{\rho N}{2-\gamma}||A_i||^2$ respectively. Calculating $||A_i||$ introduces the computational complexity of evaluating eigen values of the matrix $A_i^T A_i$. This is in contrast to the DDBi-ADMM algorithm that only requires the computation of the convex conjugate function.

4.3.1 Convergence rate

As was mentioned earlier, the proposed alternative optimization problem to the problem (4.1) is comprised of the consensus problem (4.13) and x_i-subproblems (4.14). As evaluating the solutions to the x_i-subproblems (4.14) does not require an iterative procedure, the convergence rate of the proposed approach will depend only on the iterative algorithms that address the problem (4.13). As we used the Bi-ADMM algorithm as a distributed solution to the consensus problem (4.13), the convergence rate of the proposed algorithm is the same as the Bi-ADMM algorithm. According to [202] we have the following inequalities:

$$0 \leq \sum_{i\in\nu} \sum_{j\in N(i)\setminus i} [-\text{sign}(j-i)(\bar{\delta}_{i|j}^K - \delta_{i|j}^*)^T \bar{\lambda}_j^K - (\bar{\lambda}_i^K - \lambda_i^*)^T \\ .\text{sign}(j-i)\bar{\delta}_{j|i}^K] + p(\bar{\Lambda}^K, \bar{\delta}^K) \leq O(1/K) \tag{4.19}$$

$$0 \leq \sum_{i\in\nu} \sum_{j\in N(i)\setminus i} [-\text{sign}(j-i)(\breve{\delta}_{i|j}^K - \delta_{i|j}^*)^T \breve{\lambda}_j^K - (\breve{\lambda}_i^K - \lambda_i^*)^T \\ .\text{sign}(j-i)\breve{\delta}_{j|i}^K] + p(\breve{\Lambda}^K, \breve{\delta}^K) \leq O(1/K) \tag{4.20}$$

that hold for the synchronous and asynchronous updating schemes respectively where $(\bar{\Lambda}^K, \bar{\delta}^K) = (\frac{1}{K}\sum_{k=1}^K \Lambda^k, \frac{1}{K}\sum_{k=1}^K \delta^k)$, $\Lambda = [\lambda_1^T, ..., \lambda_N^T]^T$, K denotes the total number of iterations and $p(\Lambda, \delta)$ is defined as:

$$p(\Lambda, \delta) = \sum_{i\in\nu}[h_i(\lambda_i) + h_i^*(\sum_{j\in N(i)\setminus i} \text{sign}(j-i)\delta_{i|j}) \tag{4.21}$$

For the asynchronous updating scheme, it is assumed that at each iteration node $i = \text{mod}(k, N) + 1$ is activated where mod denotes the modulo operator and the total number of iterations is $(K-1)N$, which is comprised of K segments of N iterations and $(\breve{\Lambda}^K, \breve{\delta}^K) = (\frac{1}{K}\sum_{k=1}^K \Lambda^{kN}, \frac{1}{K}\sum_{k=1}^K \delta^{kN})$.

The inequalities (4.19) and (4.20) are derived based on the variational inequalities which have been the basis for the convergence analysis of ADMM based algorithms and have been used in many papers, e.g. see [43, 74]. For detailed derivation and the proof of the above inequalities the reader is referred to [202, 204].

4.3.2 Remark

In the following we summarize the advantages and disadvantages of the proposed approach. The advantages of the proposed approach include: 1) It converges without requiring conservative movements (e.g. without requiring the under-relaxation steps as in (4.10) which imply consideration of a smaller step towards the output of ADMM). 2) It converges with the rate of $O(1/k)$. 3) It can be implemented both synchronously and asynchronously. 4) It provides a communication efficient platform for distributed optimization. The disadvantages of the proposed approach include: 1) It requires the calculation of the convex conjugate function. 2) It is not applicable when $f_i(x_i)$ is an affine function of x_i.

4.4 Experimental Results

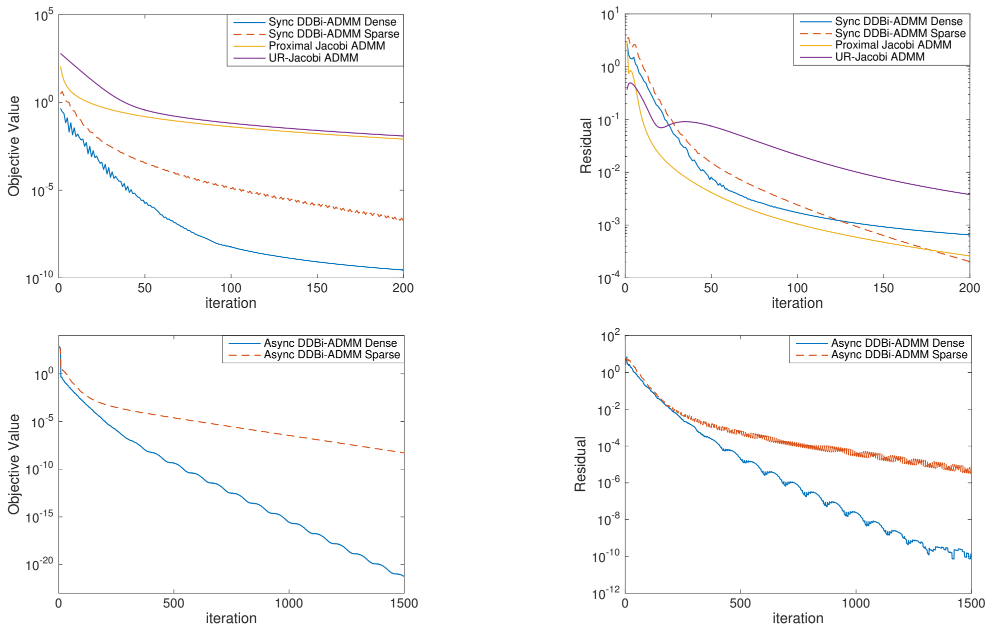
Many problems can be cast into the problems of the form (4.1). Examples include the basis pursuit problem [27,33], the exchange problem [179,180], robust principal component analysis [28, 208] and image denoising [153, 160].
To evaluate the performance of the DDBi-ADMM algorithm we implement it on the exchange problem, which is defined as:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{N} x_i = 0 \end{aligned} \tag{4.22}$$

and compare the results with the UR-Jacobi ADMM Algorithm 3 and the Proximal Jacobi ADMM Algorithm 4. In order to assess the performance of the DDBi-ADMM algorithm in terms of the consistency of agents in estimation of the variables λ_i, $i = 1, ..., N$ we added a consistency controlling parameter ζ to the update of the dual variables δ as below:

$$\delta_{i|j}^{k+1} = \delta_{j|i}^{k} + \zeta \text{sign}(j - i)(\lambda_j^k - \lambda_i^{k+1}) \tag{4.23}$$

in all of our experiments in this chapter.

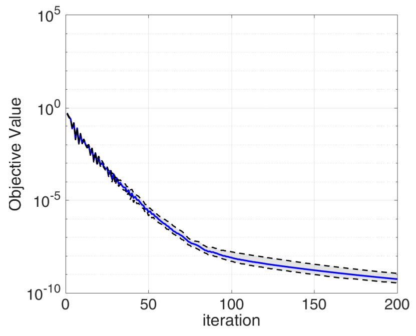


(a) Objective (b) Residual

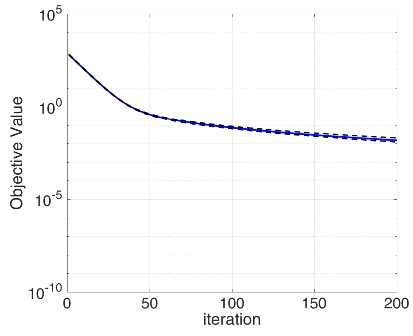
Figure 4.3: Illustration of the convergence behaviour of the DDBi-ADMM algorithm over a network with 9 agents with objective as $\sum_{i=1}^{9} f_i(x_i)$ and residual as $||\sum_{i=1}^{9} x_i||_2$.

The functions $f_i(x_i)$ were considered as $\frac{1}{2}||G_i x_i - d_i||_2^2$ where $G_i \in \mathbb{R}^{q \times n_i}$ and $d_i \in \mathbb{R}^q$. In the experiments we set $n_i = 10, \ \forall \ i = 1, ..., N$, $q = 10$ and $\zeta = 0.9$. For the Proximal Jacobi ADMM algorithm we set $P_i = \rho(\frac{N}{2-\gamma} - 1)$ and $\gamma = 1$ and for the UR-Jacobi ADMM algorithm we set $\eta = 1.99$. The latter settings are the optimal settings for the competing algorithms. The aforementioned parameter values are optimal since they provide the maximum step length towards the optimal solution. They are also the extremum values that the parameters can take such that the convergence is guaranteed [42,71]. To make a fair comparison between the algorithms we set $\rho = 1$.

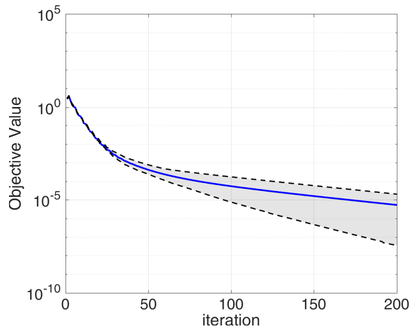
We also evaluated the performance of the algorithms in terms of the scale of the problem. Fig. 4.3 shows the average results after 20 trials for a small scale problem with $N = 9$. Fig. 4.4 shows the confidence interval of the results in Fig. 4.3. In Fig. 4.5 we plot the average results after 20 trials for a large scale problem with $N = 121$ agents. For the asynchronous case we assumed that at each iteration one agent is activated with a uniform probability distribution.



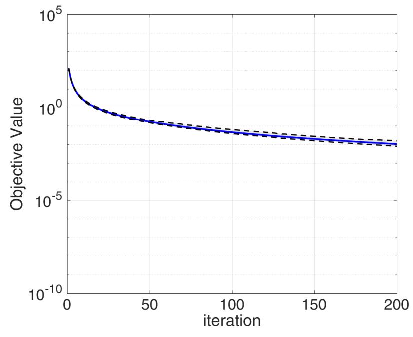
(a) Sync DDBi-ADMM Dense



(a) UR-Jacobi ADMM



(b) Sync DDBi-ADMM Sparse

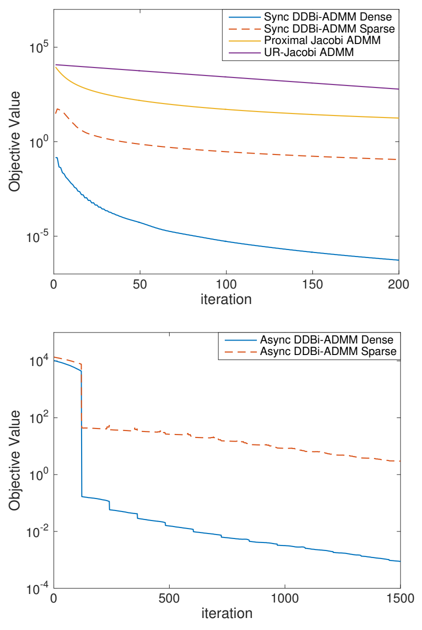


(c) Proximal Jacobi ADMM

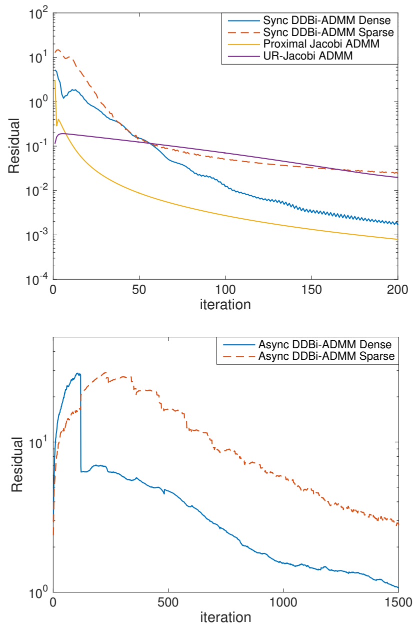
Figure 4.4: Illustration of the confidence interval of the mean of the objective value when the algorithms are applied in a synchronous updating manner on a network of 9 agents.

To asses the convergence rate of the DDBi-ADMM algorithm over sparse communication graphs, in each of the experiments we also applied the algorithm separately over a sparse graph. To derive the connectivity pattern of the sparse graphs we assumed that the agents are located on a 2 dimensional x-y plane in the form of a $\sqrt{N} \times \sqrt{N}$ grid where the distance between any two consecutive agents in the x axis and y axis is equal to l_d. We derived a communication graph by producing a link between any two agents when their distance was less than or equal to l_d.

From the results in figures 4.3 and 4.5 we see that the Sync DDBi-ADMM has superior performance in optimization of the objective function and a comparable performance in optimization of the linear constraint in comparison with the other two algorithms. This results from the fact that DDBi-ADMM exploits a Lagrangian type function (without any penalty term) for estimation of the op-



(a) Objective

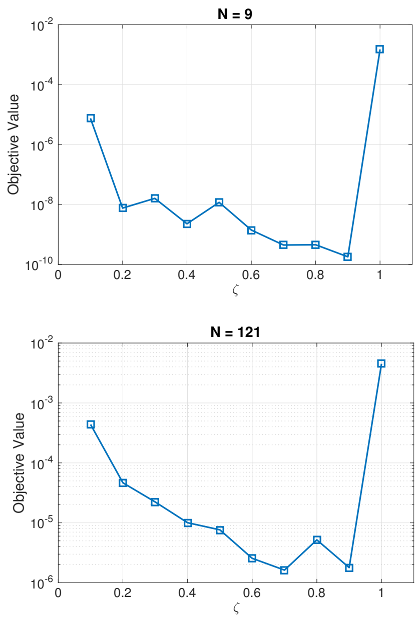


(b) Residual

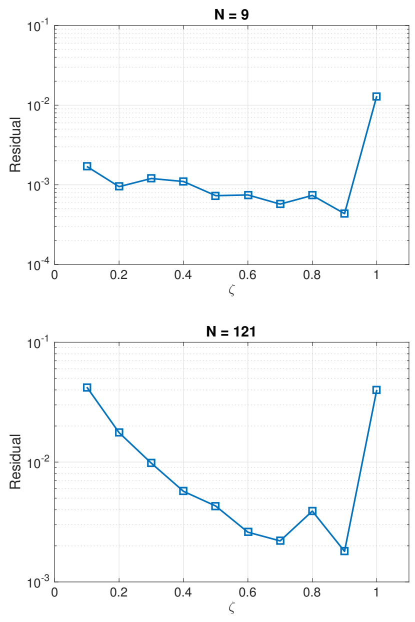
Figure 4.5: Illustration of the convergence behaviour of the DDBi-ADMM algorithm over a network with 121 agents with objective as $\sum_{i=1}^{121} f_i(x_i)$ and residual as $||\sum_{i=1}^{121} x_i||_2$.

timal primal variables, as the minimizer of the objective function, while it uses a corresponding dual function as the implicit minimizer of the residual of the primal constraint. The results in figures 4.3 and 4.5 imply that not only the proposed algorithm is superior in terms of the required communication steps but also its superior in terms of the communication overhead. This is because lower number of iterations and consequently lower number of packets and bits are transmitted to reach to a certain level of accuracy.

Fig. 4.6 shows the objective as $\sum_{i=1}^{N} f_i(x_i)$ and the residual as $||\sum_{i=1}^{N} x_i||_2$ after 200 iterations for different values of ζ using the Sync DDBi-ADMM algorithm over the dense graphs. From the results in Fig. 4.6 we see that the residual of the primal constraint as well as the objective value depends on the value of the parameter ζ and the optimal value of ζ is attained at the values other than 1, which is the case when ordinary Bi-ADMM is applied. The aforementioned dependency is the consequence of controlling the consistency of the neighbour agents in estimation of λ_is via ζ.



(a) Objective



(b) Residual

Figure 4.6: Illustration of the performance of the Sync DDBi-ADMM algorithm over the dense graphs for different ζ.

4.5 Summary

In this chapter we introduced a primal-dual algorithm called DDBi-ADMM as a solution for linearly constrained convex optimization problems where all primal variables are coupled by a global single block linear constraint. The algorithm exploits both a Lagrangian type function and an augmented Lagrangian type function and converges with $O(1/k)$ rate of convergence. The proposed algorithm can be applied both synchronously and asynchronously over any connected graph. In contrast to the traditional methods such as Jacobi ADMM, UR-Jacobi ADMM and Proximal Jacobi ADMM, the proposed algorithm is more efficient in terms of required communication per iteration. The DDBi-ADMM algorithm also provides efficiency in terms of the required transmission bandwidth in cases $m < \frac{n_i+n_j}{4}$.

In the next chapter, we investigate distributed algorithms for the constrained problems with general linear equality constraints, where the constraint is separable. In these kind of problems, each partial constraint includes a pair of decision variables that results in making a link (edge) between the nodes that hold the decision variables on the graph of the problem. We will see how we can formulate a consensus problem by a simple reformulation on the general linearly constrained problem.

5

Fully asynchronous distributed optimization

5.1 Introduction

In this chapter, we consider the following constrained optimization problem over a network of nodes denoted by the set ν with cardinality $N = |\nu|$

$$\begin{aligned} &\min_{x_1,\ldots,x_{|\nu|}} \sum_{i\in\nu} f_i(x_i) \\ &\text{s.t.} \quad A_{i\to j}x_i + A_{j\to i}x_j = c_{ij} \;\; \forall (i,j) \in \varepsilon, \end{aligned} \tag{5.1}$$

where ε denotes the set of all edges of the network with cardinality $M = |\varepsilon|$, the global objective function is $f(x) = \sum_{i\in\nu} f_i(x_i)$, $x = [x_1^T, ..., x_{|\nu|}^T]^T$, $(x_i, x_j, A_{i\to j}, A_{j\to i}, c_{ij}) \in (\mathbb{R}^{n_i}, \mathbb{R}^{n_j}, \mathbb{R}^{n_{ij}\times n_i}, \mathbb{R}^{n_{ij}\times n_j}, \mathbb{R}^{n_{ij}})$, $c_{ij} = c_{ji}$ $\forall(i,j) \in \varepsilon$, $A_{i\to j}$ and $A_{j\to i}$ are arbitrary matrices and $f_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$ is a closed, proper and convex function. The problem (5.1) arises in many massive data processing applications such as wireless communications, telecommunications and cloud learning [51, 202, 203, 209]. In these applications it is generally challenging to store the data in one location and to process the data in one single processor, and approaches towards its solution have been considered in information processing, signal processing and machine learning communities.

Because of the separability of the objective function in the problem (5.1), it can be formulated as a distributed optimization. The distributed optimization can be performed over a network of N nodes that cooperatively try to solve a global problem using local information and functions. To address the problem (5.1), it is usually first converted to the following problem [187, 188]:

$$\begin{aligned} &\min_{x,z} \sum_{i\in\nu} f_i(x_i) + I_\xi(z) \\ &\text{s.t.} \quad Dx + Hz = c, \end{aligned} \tag{5.2}$$

where $I_\xi(z) = \sum_{(i,j)\in\varepsilon} I_{\xi_{ij}}(z_{ij})$ is an indicator function of a convex set ξ defined as

$$I_\xi(z) = \begin{cases} 0 & \text{if } z \in \xi \\ +\infty & \text{otherwise} \end{cases} \tag{5.3}$$

z is a column vector that is created by stacking all $z_{ij}, (i,j) \in \varepsilon$, $z_{ij} = [z_{ij}^{i^T}, z_{ij}^{j^T}]^T$ and the convex sets ξ and $\xi_{ij} \subset \xi$ are given by

$$\begin{aligned} \xi &= \left\{z : z_{ij}^i + z_{ij}^j = 0 \ \forall(i,j) \in \varepsilon\right\} \\ \xi_{ij} &= \left\{z_{ij} : z_{ij}^i + z_{ij}^j = 0 \ (i,j) \in \varepsilon\right\} \end{aligned} \tag{5.4}$$

The constraints in (5.1) are preserved in (5.2) by enclosing their structure into the matrices D and H and the vector c as:

$$D = \begin{bmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_N \end{bmatrix}, \quad H = I, \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \tag{5.5}$$

where A_i and c_i are obtained by vertically stacking all $A_{i\to j}$ and c_{ij}, $j \in N(i)\backslash i$[1] respectively and I is an identity matrix. For example, by making use of the constraint in (5.2) and exploiting the indicator function $I_{\xi_{ij}}(z_{ij})$, the constraint

[1]$N(i)$ denotes the neighbours of node i include i itself and $N(i)\backslash i$ excludes node i from its neighbours.

on the edge (i, j) of the original problem (5.1) can be represented as:

$$A_{i\rightarrow j}x_i + z_{ij}^i = \frac{c_{ij}}{2} \tag{5.6}$$

$$A_{j\rightarrow i}x_j + z_{ij}^j = \frac{c_{ij}}{2} \tag{5.7}$$

By summing (5.6) with (5.7) we obtain the same constraint at the edge (i, j) of the original problem.

The aforementioned conversion is motivated by the ability to use the ADMM algorithm as a distributed algorithm that alternates between optimizing only two variables x and z. The convergence of the ADMM algorithm is only guaranteed over the problems with two blocks of variables [32,54]. By applying the ADMM algorithm over problem (5.2), one can still enjoy the separability of $f(x)$ when one optimizes the variable x by allocating the optimization of x_i to the i^{th} node of the network. However, the introduced auxiliary variable z in (5.2) requires the existence of a local master node, for each pair of the worker nodes (ν_i, ν_j) that are connected by the edge $(i, j) \in \varepsilon$, to handle the optimization of the local variable z_{ij}. This not only increases the computational complexity but also requires a change on the graphic structure of the network. Therefore it is more desirable to develop distributed algorithms for the problems of the form (5.1) rather than (5.2) since it does not include the auxiliary variable z. Nevertheless many applications of (5.2) exist. Examples are distributed consensus optimization problems such as demodulation [209], diffusion adaptation [167] and dual averaging [46].

Literature review

In a great number of recent studies, researchers put their effort in finding a distributed solution for problems of the form (5.1) by using (5.2) rather than using (5.1) directly. However, many of the proposed solutions have limitations. One such limitation is the sublinear rate, $O(1/\sqrt{k})$ where k denotes the iteration counter, of convergence of subgradient methods [23,136,155,167] that exploit the Lagrangian function. The latter limitation was addressed by using the *augmented* Lagrangian function and applying the alternating direction method of multipliers (ADMM) [24]. ADMM is a special case of the Douglas-Rachford splitting method [47, 116], which has the best known convergence rate $O(1/k)$ for the general convex problems. Basic ADMM is also subject to limitations. For example it requires synchronous updating of the nodes, which makes it impractical in real-world scenarios when coordination of a set of distant agents is difficult.

To reach to the solution of constraint problems, e.g. (5.2), one can start from an initial value for the primal variables, (x,z), and Lagrange multipliers (dual variables) and then iteratively update their values. Algorithms with asynchronous updating are often more practical than algorithms with synchronous updating. ADMM with asynchronous updating for problems that exploit the auxiliary variable z, e.g. (5.2), is introduced in [90,188,206]. The problems that are considered in [90,188,206] are consensus-based optimization problems that are a special case of (5.2). In [188] a local variable z_{ij} is allocated to each pair of nodes (ν_i, ν_j) that are connected by the edge $(i, j) \in \varepsilon$ while in [90] the nodes are partitioned into L subsets $B_\jmath \subset \nu$, $\jmath = 1, ..., L$ with cardinality $2 \leq |B_\jmath| < N$ and a local variable $z_\jmath$ is allocated to each subset $B_\jmath$. In fact [90] is a generalized version of [188] and they are equivalent to each other for the special case of $|B_\jmath| = 2\ \forall \jmath$. The asynchrony in [90, 188] is achieved by activating an edge or subset of edges in [188] or activating a subset or subsets of the nodes in [90], of the original graph G, at a time and updating their local variables x_i's and the local variable z_{ij} or $z_\jmath$ of their local master node in an alternating manner. The drawback of the algorithms of [90, 188] is that, although there is no requirement for a global clock to coordinate the update of all the nodes, there is the need for a local clock for each set of nodes that are connected to a local master node. [206] eliminated the need for any clock by making use of a partial barrier mechanism [6] in which the master node updates its variable after it receives at least one update from its allocated worker nodes. However, in [206], in contrast to [90,188], a global variable z (a global master node) is applied which conflicts with the de-centralized nature of the problem. It is good to note that the distributed asynchronous algorithms of [90,188,206] also have the computational complexity of two minimizations and one gradient step per iteration: optimization of the primal variable x and auxiliary variable z and a gradient step towards the optimum point of the Lagrange multiplier.

Full asynchronism

In the asynchronous Bi-ADMM [202] algorithm, the computational complexity of evaluation of the variable z is reduced by considering the problem of the form (5.1). By considering the problem of the form (5.1) they could also preserve the graphic structure of the network by eliminating the need for local or global master node(s). To be able to eliminate the need for any local clock, [202] splits the local dual variable associated to each local constraint by introducing two copies of it and allocating the update of each copied version separately to the nodes that are connected via that local constraint. This allows the development of a fully asynchronous distributed algorithm in the sense that at each iteration one node or a subset of nodes can be activated without the need for any coordination. To build consistency between the copied versions of the local dual variables, a constrained dual problem is formulated. The constrained dual problem in Bi-ADMM algorithm results in the formulation of a primal-dual augmented Lagrangian function that requires the additional complexity of evaluating conjugate functions.

Contribution

In this chapter, we consider constrained problems of the form (5.1) and convert it to an unconstrained optimization problem with a new objective function that is simpler than the primal-dual augmented Lagrangian function that is introduced in [202]. By making use of the penalty method [52,120,172], the proposed objective function is formed by adding a penalty term to the augmented Lagrangian function, which forms a regularized augmented Lagrangian function. Similar to the primal-dual augmented Lagrangian used in Bi-ADMM [202], the proposed regularized augmented Lagrangian allows the activation of one node or a subset of nodes per iteration. This consequently leads to a fully asynchronous algorithm without the need for any local or global master node(s) and any clock for coordination. However, in contrast to the primal-dual augmented Lagrangian function that is introduced in [202], the proposed regularized augmented Lagrangian function does not require calculation of the conjugate function. By exploiting the problems of the form (5.1) rather than (5.2), the proposed method has the advantage of low computational complexity by eliminating the update of the variable z in comparison with the basic ADMM-based algorithms [90,188,206] and allows the preservation of the graphic structure of the network similar to [202].

In order to decouple the update of dual variable λ_{ij} associated to the constraint on the edge (i,j), similar to [202] we introduce two copies of the dual variable as $\lambda_{i|j}$ and $\lambda_{j|i}$, where $\lambda_{i|j}$ is the variable that is held at node i and $\lambda_{j|i}$ is the variable that is held at node j. However, in contrast to [202], instead of the formulation of an augmented Lagrangian function for the dual problem we add a penalty function in the form of $||\lambda_{i|j} - \lambda_{j|i}||_2^2$ to the augmented Lagrangian of the primal problem to relax the assumption that $\lambda_{i|j}=\lambda_{j|i}$. Since the constraint on the equality of $\lambda_{i|j}=\lambda_{j|i}$ is approximated, as a consequence of using a penalty function, the inaccuracy in such an approximation may affect the final solution of the problem. Therefore, we provide a measure of the inaccuracy in such an approximation on the optimal value of the objective function.

The added penalty term allows the decoupling of the update of the dual variable λ_{ij} and associates the update of $\lambda_{i|j}$ and $\lambda_{j|i}$ separately with the node i and j. This results in activating the nodes independently, similar to but in a simpler way than [202], in which the updates of the primal and dual variables can take place separately in each node. Existence of the penalty function also results in a strongly concave regularized augmented Lagrangian function in terms of the dual variables for fixed primal variables. The strong concavity of the regularized augmented Lagrangian function converts the gradient step towards the optimum point of the dual variables to a maximization step at each node. This makes the optimization procedure of the dual variables more stable than the gradient optimization, since they will have a finite and closed-form solution.

The remainder of this chapter is organized as follows. In Section 5.2 we briefly discuss the existing synchronous and asynchronous distributed algorithms that have been used in the literature. In Section 5.3 we explain the proposed asynchronous distributed algorithm. The evaluation of the proposed asynchronous algorithm is presented in Section 5.4. It is followed by a conclusion in Section 5.5.

5.2 Review of distributed ADMM-based algorithms

In this section, we review the synchronous and asynchronous distributed ADMM-based algorithms that were introduced to address the consensus-based optimization problems by making use of the problems of the form (5.2). In Section 5.2.1 we discuss the synchronous algorithm followed by a discussion on the asynchronous algorithm in Section 5.2.2.

5.2.1 Synchronous distributed consensus ADMM

In distributed synchronous consensus ADMM-based methods the constraint problem 5.2 can be reduced to:

$$\begin{aligned} &\min_{x_1,\dots,x_{|\nu|},z} \sum_{i\in\nu} f_i(x_i) \\ &\text{s.t.} \quad x_i - z = 0, \quad \forall i \in \nu. \end{aligned} \tag{5.8}$$

To convert the above problem into an unconstrained problem, the augmented Lagrangian function is used as an objective function:

$$L_\rho(x,z,\lambda) = \sum_{i\in\nu}\{f_i(x_i) + \lambda_i^T(x_i - z) + \frac{\rho}{2}||x_i - z||_2^2\}, \tag{5.9}$$

where $\rho > 0$ and is the penalty parameter, z and λ_i are the auxiliary (consensus) variable and the i^{th} dual variable respectively and $\lambda = \{\lambda_i | i \in \nu\}$.
According to [24], by making use of the augmented Lagrangian function in (5.9), the solution for the problem (5.8) can be obtained via the ADMM algorithm as:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i}(f_i(x_i) + \frac{\rho}{2}||x_i - z^k + (\lambda_i^k/\rho)||_2^2), \;\; \forall i \in \nu \\ z^{k+1} &= \arg\min_{z}(\sum_{i\in\nu}\frac{\rho}{2}||x_i^{k+1} - z + (\lambda_i^k/\rho)||_2^2) \\ \lambda_i^{k+1} &= \lambda_i^k + \rho(x_i^{k+1} - z^{k+1}), \;\; \forall i \in \nu \end{aligned} \tag{5.10}$$

where k is the iteration number. The algorithm in (5.10) is referred to as *alternating* direction method of multipliers since there is an alternation between the optimization of the variables x_i's and z.
The updates in (5.10) can be carried out in a distributed processing manner by assigning the updates of x_i and λ_i over a set of $N = |\nu|$ worker nodes and the updates related to z to a global master node over a communication graph with star topology. However, the z update (5.10) requires the updates of all $x_i, i = 1, ..., N$ in the worker nodes to be done before the update in the master node takes place. This leads to the need for a global clock to coordinate the updates of the master and worker agents that form a synchronous updating scheme. In large-scale problems and especially when there is a set of disparate agents with different processing speeds, the synchronous update becomes a burden. In the following we refer to (5.10) as Sync-ADMM.

5.2.2 Asynchronous distributed consensus ADMM

To eliminate the need for coordination of all the nodes of a network and the need for a global master node, in [188] and [90] an asynchronous de-centralized version of the ADMM algorithm was introduced. By partitioning the set of all the nodes into smaller subsets and allocating a local consensus variable to each subset, at each iteration only an arbitrary subset of the nodes along with their local master node, which holds the local consensus variable, are activated. This allows elimination of the need for a *global* clock. However, since there is the need for an alternation between optimization of the primal variables of the active worker nodes and the local consensus variable of their local master node, there is the need for a local clock to coordinate their updates.

The Async-ADMM algorithm [90] introduces a subset $B \subset \nu$ and considers a collection of $B_1, ..., B_L$ such that $\bigcup_{\jmath=1}^{L} B_\jmath = \nu$ and assumes that at each iteration one subset $B_\jmath$ is activated. The updates of Async-ADMM can be summarized as follows:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i}(f_i(x_i) + \frac{\rho}{2}||x_i - z_\jmath^k + (\lambda_{i,\jmath}^k/\rho)||_2^2), \ i \in B_\jmath \\ z_\jmath^{k+1} &= \arg\min_{z}(\sum_{i=1}^{|B_\jmath|} \frac{\rho}{2}||x_i^{k+1} - z + (\lambda_{i,\jmath}^k/\rho)||_2^2) \\ \lambda_{i,\jmath}^{k+1} &= \lambda_{i,\jmath}^k + \rho(x_i^{k+1} - z_\jmath^{k+1}), \end{aligned} \tag{5.11}$$

where $\lambda_{i,\jmath} = \{\lambda_i | i \in B_\jmath\}$ and $z_\jmath$ is the local consensus variable. The second minimization together with the gradient step in equation (5.11) leads to the need for a local clock.

5.3 Fully asynchronous distributed algorithm

In this section we discuss a straightforward but novel derivation of a fully asynchronous distributed algorithm that eliminates the need for any coordination across the nodes. We first derive an algorithm for the general problem (5.1) in Section 5.3.1 and provide a convergence proof in Section 5.3.2. In Section 5.3.3 we introduce an efficient synchronous update that can only be applied to synchronous systems followed by the derivation of a fully asynchronous algorithm for the consensus-based problems in Section 5.3.4.

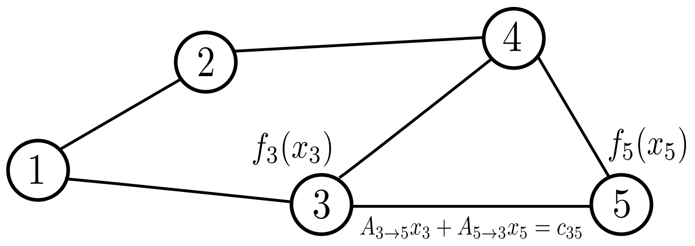


Figure 5.1: Illustration of the graph of the problem (5.1) for a sample network of 5 nodes.

5.3.1 The general problem

To eliminate the need for any master node and to reduce the computational complexities that are introduced by considering the auxiliary variable z, we consider the problem of the form (5.1). The problem (5.1) can be considered as a general problem since it includes arbitrary matrices $A_{i\to j}s$. In this section we discuss in detail how the general problem can be formulated as a consensus problem by using special forms of $A_{i\to j}s$. Fig 5.1 illustrates a representative graph of this problem. According to the graphical structure of the problem, the augmented Lagrangian of the problem for a graph G=(ν,ε) with $|\nu| = N$ can be written as:

$$\begin{aligned} L_\rho(x,\lambda) =& \sum_{i\in\nu} f_i(x_i) + \sum_{(i,j)\in\varepsilon} \lambda_{ij}^T(c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j) \\ &+ \sum_{(i,j)\in\varepsilon} \frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j||_2^2, \end{aligned} \tag{5.12}$$

where $\lambda = \{\lambda_{ij}|(i,j)\in\varepsilon\}$.

Assuming that the KKT conditions [96, 97, 104] hold for the problem (5.1), the solution of the problem can be evaluated at the saddle point (x^*,λ^*) of the augmented Lagrangian (5.12) by following a min-max procedure

$$(x^*,\lambda^*) = \arg\max_{\lambda}\min_{x} L_\rho(x,\lambda). \tag{5.13}$$

To benefit from the separability of the objective function, $f(x)$, in evaluation of (x^*,λ^*) from (5.13), one can arbitrarily choose an initial value for (x,λ) and update their values iteratively by making use of the min-max optimization procedure in (5.13) as below

$$\begin{aligned} x_i^{k+1} =& \arg\min_{x_i}(f_i(x_i) - \sum_{j\in N(i)\backslash i} \lambda_{ij}^{k^T}(A_{i\to j}x_i) \\ &+ \sum_{j\in N(i)\backslash i} \frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j^k||_2^2),\ i\in\nu \end{aligned} \tag{5.14}$$

$$\lambda_{ij}^{k+1} = \lambda_{ij}^k + \rho(c_{ij} - A_{i\to j}x_i^k - A_{j\to i}x_j^k),\ j\in N(i)\backslash i \tag{5.15}$$

where $N(i)$ denotes the neighbours of node i include i itself and $N(i)\backslash i$ excludes node i from its neighbours. In (5.15) a gradient ascent step is used for optimization of the dual variables since $L_\rho(x,\lambda)$ is an affine function of λ for any given values of x and does not have a maximum. The gradient step in (5.15) is guaranteed to approach iteratively to the maximum of the dual function $g(\lambda)$, see for example [21, 24], that is defined as

$$g(\lambda) = \min_x L_\rho(x,\lambda). \tag{5.16}$$

Although by using the updates (5.14)-(5.15) we are one step closer to elimination of the need for coordination between the nodes as a consequence of eliminating the variable z, the dual update (5.15) requires access to the simultaneous primal updates x_i^k and x_j^k which brings the need for a local clock to coordinate the updates of the neighbour nodes i and j. It also suggests the existence of a local master node for any pair of nodes (ν_i,ν_j), that are connected, to handle the optimization of their local dual variable λ_{ij}. To address this problem we decouple the dual update (5.15) by introducing two copies of the dual variable λ_{ij} as $\lambda_{i|j}$ and $\lambda_{j|i}$ with the constraint that $\lambda_{i|j} = \lambda_{j|i}$. We then add a quadratic penalty term in the form of $||\lambda_{i|j} - \lambda_{j|i}||_2^2$ to the augmented Lagrangian, as an application of the penalty method [52, 120, 172] to the method of multipliers [76, 127]. This penalizes the violation of the constraint, $\lambda_{i|j} = \lambda_{j|i}$, which leads to the following regularized augmented Lagrangian function:

$$\hat{L}_{\rho,\alpha}(x,\lambda) = \sum_{(i,j)\in\varepsilon} \hat{L}_{\rho,\alpha}^{ij}(x_i,x_j,\lambda_{i|j},\lambda_{j|i}), \tag{5.17}$$

where $\lambda = \{\lambda_i | i\in\nu\}$, $\lambda_i = \{\lambda_{i|j} | j\in N(i)\backslash i\}$ and $\hat{L}_{\rho,\alpha}^{ij}(x_i,x_j,\lambda_{i|j},\lambda_{j|i})$ is the regularized augmented Lagrangian function associated with the constraint on the edge (i,j), which is perceived by the node i as $\hat{L}^{(ij),i}_{\rho,\alpha}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i})$ defined as

$$\begin{aligned}\hat{L}^{(ij),i}_{\rho,\alpha}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i}) = &\frac{1}{v_i} f_i(x_i) + \lambda^T_{j|i}(c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j) \\ &+ \frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j||^2_2 - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||^2_2,\end{aligned} \tag{5.18}$$

where $v_i = |N(i)\backslash i|$, $\frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j||^2_2$ is the penalty term (the augmentation) that is related to the constraint, $A_{i\to j}x_i + A_{j\to i}x_j = c_{ij}$, of the primal problem and $\frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||^2_2$ is the added regularization term for restricting the violation of the constraint $\lambda_{i|j} = \lambda_{j|i}$.

$\hat{L}^{ij}_{\rho,\alpha}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i})$ is also perceived by the node j as $\hat{L}^{(ij),j}_{\rho,\alpha}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i})$ defined as

$$\begin{aligned}\hat{L}^{(ij),j}_{\rho,\alpha}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i}) = &\frac{1}{v_j} f_j(x_j) + \lambda^T_{i|j}(c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j) \\ &+ \frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j||^2_2 - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||^2_2.\end{aligned} \tag{5.19}$$

It should be noted that α needs to be a positive value ($\alpha > 0$) since otherwise the penalty term, $\frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||^2_2$, will not have a minimum. To avoid confusion we refer to ρ and α as primal and dual penalty parameters respectively.

Throughout this chapter we make the following assumption:

Assumption 1: *The regularized unaugmented Lagrangian function, $\hat{L}_{0,\alpha}$, has a saddle point $u^* = (x^*, \lambda^*)$. Namely, (x^*, λ^*) satisfies the KKT conditions*:

$$\begin{aligned}&\sum_{j\in N(i)\backslash i} A^T_{i\to j}\lambda^*_{j|i} \in \partial f_i(x^*_i) \quad ,\forall i \in \nu \\ &A_{i\to j}x^*_i + A_{j\to i}x^*_j = c_{ij} \quad ,\forall (i,j) \in \varepsilon.\end{aligned} \tag{5.20}$$

Assuming that the Assumption 1 is satisfied, we can evaluate the solution of the problem (5.1), by making use of the regularized augmented Lagrangian (5.17), through the following min-max procedure:

$$\begin{aligned}x^{k+1}_i &= \arg\min_{x_i} \sum_{(i,j)\in\varepsilon} \hat{L}^{(ij),i}_{\rho,\alpha}(x_i, x^k_j, \lambda^k_{j|i}),\ i \in \nu \\ \lambda^{k+1}_i &= \arg\max_{\lambda_i} \sum_{(i,j)\in\varepsilon} \hat{L}^{(ij),j}_{\rho,\alpha}(x^{k+1}_i, x^k_j, \lambda^k_{j|i}),\ i \in \nu\end{aligned} \tag{5.21}$$

where node i is responsible for the update of x_i and λ_i.

The updates in (5.21) can be run both synchronously and asynchronously over a network of N agents. In the synchronous updating scheme at each iteration all the nodes are activated and update the variables x_i and λ_i. By partitioning the nodes into L subsets $C_\jmath \subset \nu$, $\jmath = 1, ..., L$ with cardinality $1 \leq |C_\jmath| < N$ where $\bigcup_{\jmath=1}^{L} C_\jmath = \nu$, in the asynchronous updating scheme only a subset $C_\jmath$ of nodes are activated at each iteration and receive the new updates. It is important to note that the distinct subsets $C_\jmath$s are not necessarily disjoint and the probability of activation of each subset must be greater than zero.

Using (5.21) leads to the following updates for each node:

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i}\{f_i(x_i) + \sum_{j\in N(i)\backslash i} [-\lambda_{j|i}^{k^T}(A_{i\to j}x_i) \\ &+ \frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j^k||_2^2]\},\ i \in \nu \end{aligned} \tag{5.22}$$

$$\begin{aligned} \lambda_{i|j}^{k+1} &= \arg\max_{\lambda_{i|j}}\{\lambda_{i|j}^T(c_{ij} - A_{i\to j}x_i^{k+1} - A_{j\to i}x_j^k) \\ &\qquad -\frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}^k||_2^2\},\ j \in N(i)\backslash i. \end{aligned} \tag{5.23}$$

Algorithm 7 shows the proposed asynchronous algorithm where $\bar{C}_\jmath$ denotes the complement of set $C_\jmath$. By comparing the proposed algorithm with the Bi-ADMM algorithm [202] we see that x_i update (5.22) is similar to the x_i update in [202, eq. (8)] for $\rho = 1$ while the dual variables in (5.23) are updated without the need of calculating a conjugate function in comparison with [202, eq. (9)].

As was mentioned earlier, the inaccuracy in approximation of the constraint $\lambda_{i|j} = \lambda_{j|i}$ may affect the final solution of the problem. In order to measure the effect of such an approximation on the optimal value of the objective function and the residual of the constraint in (5.1), without loss of generality, we consider a connected graph of two nodes (N=2) that is comprised of node i and node j. For the purpose of simplicity, we then compare the *unaugmented* Lagrangian function, $L_0(x, \lambda)$, in (5.12) with the regularized *unaugmented* Lagrangian function, $\hat{L}_{0,\alpha}(x, \lambda)$, in (5.17). Accordingly, we introduce the following notations and functions. We denote by δ_{ij} the difference between the optimal dual variables $\lambda_{i|j}^*$ and $\lambda_{j|i}^*$ as $\delta_{ij} = \lambda_{i|j}^* - \lambda_{j|i}^*$. We use $\hat{g}_i(\lambda_{i|j}, \lambda_{j|i})$ to denote the dual function that is perceived by the node i, which is evaluated as:

$$\begin{aligned}\hat{g}_i(\lambda_{i|j}, \lambda_{j|i}) &= \min_{x_i} \hat{L}_{0,\alpha}^{(ij),i}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i}) \\ &= g_i(\lambda_{j|i}) - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||_2^2\end{aligned} \tag{5.24}$$

where $g_i(\lambda_{j|i})$ is defined as:

$$g_i(\lambda_{j|i}) = \min_{x_i}\{f_i(x_i) - \lambda_{j|i}^T(A_{i\to j}x_i)\} \tag{5.25}$$

We also use $\hat{g}_j(\lambda_{i|j}, \lambda_{j|i})$ to denote the dual function that is perceived by the node j, which is evaluated as:

$$\begin{aligned}\hat{g}_j(\lambda_{i|j}, \lambda_{j|i}) &= \min_{x_j} \hat{L}_{0,\alpha}^{(ij),j}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i}) \\ &= g_j(\lambda_{i|j}) - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||_2^2\end{aligned} \tag{5.26}$$

where $g_j(\lambda_{i|j})$ is defined as:

$$g_j(\lambda_{i|j}) = \min_{x_j}\{f_j(x_j) - \lambda_{i|j}^T(A_{j\to i}x_j)\} \tag{5.27}$$

Finally, we denote by $\hat{g}_{ij}(\lambda_{i|j}, \lambda_{j|i})$ the global dual function, the function that is derived from the global regularized Lagrangian function $\hat{L}_{0,\alpha}(x_i, x_j, \lambda_{i|j}, \lambda_{j|i})$, which is represented as:

$$\hat{g}_{ij}(\lambda_{i|j}, \lambda_{j|i}) = g_i(\lambda_{j|i}) + g_j(\lambda_{i|j}) - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||_2^2 \tag{5.28}$$

We now derive the global dual function, $g_{ij}(\lambda_{ij})$, from the original Lagrangian function, $L_0(x_i, x_j, \lambda_{ij})$, which is represented as:

$$\begin{aligned}g_{ij}(\lambda_{ij}) &= \min_{x_i,x_j} L_0(x_i, x_j, \lambda_{ij}) \\ &= \min_{x_i,x_j}\{f_i(x_i) + f_j(x_j) + \lambda_{ij}^T(c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j)\} \\ &= \min_{x_i}\{f_i(x_i) - \lambda_{ij}^T(A_{i\to j}x_i)\} + \min_{x_j}\{f_j(x_j) - \lambda_{ij}^T(A_{j\to i}x_j)\} \\ &= g_i(\lambda_{ij}) + g_j(\lambda_{ij})\end{aligned} \tag{5.29}$$

where the last equality in (5.29) is achieved by making use of the definitions in (5.25) and (5.27).

It should be noted that the dual function $g_{ij}(\lambda_{ij})$ is a concave function [25, Chapter 5], with its maximum is attained at λ_{ij}^*. By substituting λ_{ij}^* into (5.29) we reach the following relationship:

$$\begin{aligned} g_{ij}(\lambda_{ij}^*) &= g_i(\lambda_{ij}^*) + g_j(\lambda_{ij}^*) \\ &\leq g_i(\lambda_{j|i}^*) + g_j(\lambda_{i|j}^*) \end{aligned} \tag{5.30}$$

By introducing Δ as $\Delta = g_{ij}(\lambda_{ij}^*) - (g_i(\lambda_{j|i}^*) + g_j(\lambda_{i|j}^*))$, we will have

$$\begin{aligned} \Delta = 0 \qquad &\text{if} \qquad \delta_{ij} = 0, \\ \Delta < 0 \qquad &\text{if} \qquad \delta_{ij} \neq 0. \end{aligned} \tag{5.31}$$

The proof of the inequality in (5.30) and correctness of the conditions in (5.31) can be found in the Appendix A. By making use of the inequality in (5.30) and the equality in (5.28), we establish the following relationship

$$\begin{aligned} \hat{g}_{ij}(\lambda_{i|j}^*, \lambda_{j|i}^*) &= g_i(\lambda_{j|i}^*) + g_j(\lambda_{i|j}^*) - \frac{1}{(2\alpha)}||\lambda_{i|j}^* - \lambda_{j|i}^*||_2^2 \\ &\geq g_{ij}(\lambda_{ij}^*) - \frac{1}{(2\alpha)}||\lambda_{i|j}^* - \lambda_{j|i}^*||_2^2, \end{aligned} \tag{5.32}$$

which can be reduced to the following equality:

$$\hat{g}_{ij}(\lambda_{i|j}^*, \lambda_{j|i}^*) = g_{ij}(\lambda_{ij}^*) - \frac{1}{(2\alpha)}||\lambda_{i|j}^* - \lambda_{j|i}^*||_2^2 - \Delta. \tag{5.33}$$

By making use of the optimality conditions in (5.20), we can apply the strong duality on $\hat{g}_{ij}(\lambda_{i|j}^*, \lambda_{j|i}^*)$, which leads to the following relationship:

$$\begin{aligned} P^*(\delta_{ij}) &= \hat{g}_{ij}(\lambda_{i|j}^*, \lambda_{j|i}^*) \\ &= g_{ij}(\lambda_{ij}^*) - \frac{1}{(2\alpha)}||\lambda_{i|j}^* - \lambda_{j|i}^*||_2^2 - \Delta \\ &= P^*(0) - \frac{1}{(2\alpha)}||\delta_{ij}||_2^2 - \Delta. \end{aligned} \tag{5.34}$$

where $P^*(\delta_{ij})$ and $P^*(0)$ are the optimal value of the objective function that are attained when the regularized (augmented) Lagrangian function in (5.17) and the original (augmented) Lagrangian function in (5.12) are used respectively. By setting δ_{ij} to zero in the above equation we reach the following equality

$$P^*(0) = P^*(0). \tag{5.35}$$

From the equality in (5.34), we see that $P^*(\delta_{ij})$ can be even less than $P^*(0)$, $P^*(\delta_{ij}) < P^*(0)$, if the following inequality holds:

$$\Delta > \frac{-1}{(2\alpha)}||\delta_{ij}||_2^2. \tag{5.36}$$

In other words, with the assumption that the condition in (5.36) is met, the inaccuracy in satisfying the constraint $\lambda_{i|j} = \lambda_{j|i}$ not only does not increase the value of the objective function but also decreases it. However, such an approximation increases the residual of the constraint $A_{i\to j}x_i + A_{j\to i}x_j = c_{ij}$. To measure the residual of the constraint as a function of δ_{ij}, we consider the optimization in (5.23), which has a closed-form solution as

$$\lambda_{i|j}^{k+1} = \lambda_{j|i}^{k} + \alpha(c_{ij} - A_{i\to j}x_i^{k+1} - A_{j\to i}x_j^{k}). \tag{5.37}$$

Accordingly, at the optimal values of the dual and primal variables we have the following equality:

$$\lambda_{i|j}^{*} - \lambda_{j|i}^{*} = \alpha(c_{ij} - A_{i\to j}x_i^{*} - A_{j\to i}x_j^{*}), \tag{5.38}$$

thus, the residual of the constraint as a function of δ_{ij} is evaluated as:

$$\begin{aligned} R(\delta_{ij}) &= \frac{\lambda_{i|j}^{*} - \lambda_{j|i}^{*}}{\alpha} \\ &= \frac{\delta_{ij}}{\alpha}. \end{aligned} \tag{5.39}$$

As the proposed algorithm is derived by adding a penalty term, in terms of the dual variable, to the method of multipliers, we refer to it as asynchronous Augmented Method of Multipliers (Async-AMM) algorithm.

It is good to note that although the updates in (5.22)-(5.23) can be run both synchronously and asynchronously, the synchronous updates (5.14)-(5.15) can be reformulated to simpler synchronous updates, which we discuss in more details in Section 5.3.3.

Algorithm 7 Fully Asynchronous Distributed Augmented Method of Multipliers algorithm (Async-AMM).

```
Randomly initialize $x_i^0$ and $\lambda_i^0$ $\forall i \in \nu,\ k \leftarrow 0$
repeat
    Find the active set $C_\jmath$
    for node $i \in C_\jmath$ do
        Update $x_i$ using equation (5.22)
        Update $\lambda_i$ using equation (5.23)
    end for
    $(x_j^{k+1}, \lambda_j^{k+1}) = (x_j^k, \lambda_j^k) \quad \forall\, j \in \bar{C}_\jmath$
    $k \leftarrow k+1$
until a stopping criteria is met
```

5.3.2 Convergence proof

Before presenting the convergence proof, we first introduce the following notations. We denote by A_i, λ_i, y_i and c_i the matrix and vectors that are obtained by vertically stacking all $A_{i\to j}$, $\lambda_{i|j}$, x_j and c_{ij}, $j \in N(i)\backslash i$ respectively. We use B_i to denote a block diagonal matrix, possibly non-square, whose main diagonal is comprised of all $A_{j\to i}$, $j \in N(i)\backslash i$. We denote by A and B as block diagonal matrices, possibly non-square, whose main diagonals are comprised of all A_i and B_i, $i \in \nu$ respectively. Finally we let λ, y and $\bar{c}$ denote the vectors that are obtained by vertically stacking all λ_i, y_i and c_i, $i \in \nu$ respectively.

To evaluate the convergence of the AMM algorithm, without loss of generality, we assume that all the nodes are activated for each iteration. Given the aforementioned assumption, the first-order optimality conditions of the optimization problems in (5.22) and (5.23) are written by the following variational inequalities [50], which are a special form of the KKT conditions (5.20):

$$\begin{aligned}&f_i(x_i) - f_i(x_i^{k+1}) + \\ &(x_i - x_i^{k+1})^T \{ \sum_{j\in N(i)\backslash i} -A_{i\to j}^T [\lambda_{i|j}^k + \alpha(c_{ij} - A_{i\to j}x_i^k - A_{j\to i}x_j^{k+1})]\} \geq 0, \forall i \in \nu.\end{aligned} \tag{5.40}$$

(5.40) can be written in a compact form as:

$$\begin{aligned}&f_i(x_i) - f_i(x_i^{k+1}) + \\ &(x_i - x_i^{k+1})^T \{ -A_i^T [\lambda_i^k + \alpha(c_i - A_i x_i^k - B_i y_i^{k+1})]\} \geq 0, \forall i \in \nu.\end{aligned} \tag{5.41}$$

Summing (5.41) over all $i \in \nu$ leads to the following inequality

$$\begin{aligned} &f(x) - f(x^{k+1}) + \\ &(x - x^{k+1})^T \{-A^T[\lambda^k + \alpha(\bar{c} - Ax^k - By^{k+1})]\} \geq 0, \end{aligned} \tag{5.42}$$

which is a subset of the first-order optimality conditions of the following scheme:

$$\begin{aligned} y^{k+1} &= \arg\min_y \tilde{L}_\alpha(y, x^k, \lambda^k) \\ x^{k+1} &= \arg\min_x \tilde{L}_\alpha(y^{k+1}, x, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \alpha(\bar{c} - Ax^{k+1} - By^{k+1}) \end{aligned} \tag{5.43}$$

where $\tilde{L}_\alpha(y, x, \lambda) = f(x) + \lambda^T(\bar{c} - Ax - By) + \frac{\alpha}{2}||\bar{c} - Ax - By||_2^2$.

In fact (5.43) is an application of the original ADMM to two blocks of primal variables, for which convergence was proven in [54]. Since the condition (5.42) is met, the iterative updates (5.22)-(5.23) are convergent. Therefore, we can conclude that there is no necessity for applying the auxiliary variable z for the problems of the form (5.1), with separable constraints where each constraint only includes a weighted sum of two primal variables, for convergence purposes. Indeed, applying the auxiliary variable z is only meaningful for problems in which the constraint includes a weighted sum of $N > 2$ blocks of primal variables and is referred to as variable splitting method, see for example [22,24,187]. It is good to note the role of the dual penalty parameter α in (5.37) and (5.40). Since the dual penalty parameter α plays the role of the primal penalty parameter ρ in (5.15), as the appropriate learning rate for the update of the dual variable [24], its acceptable range of values is limited to $0 < \alpha \leq \rho$.

5.3.3 An alternative synchronous update

We saw that by exploiting the regularized augmented Lagrangian function (5.17) not only an asynchronous updating scheme but also a synchronous updating scheme can be made by activating all the nodes per iteration and using the updates in (5.22)-(5.23). In this section we aim to introduce new synchronous updates to be used as an alternative to the updates (5.22)-(5.23) in synchronous systems, because of their simplicity.

Since in synchronous optimizations there is no need to decouple the update of the dual variables by splitting the dual variables, they would not have the need to add a penalty term to the augmented Lagrangian function. Therefore, one can directly exploit the augmented Lagrangian function (5.12) for the problems of the form (5.1). As we saw earlier, making use of the augmented Lagrangian function (5.12) led to the updates in (5.14)-(5.15). To simplify (5.14)-(5.15), we consider $\sum_{j\in N(i)\backslash i}\lambda_{ij}^{k^T}A_{i\to j}$ as the data that is needed at node i at the k^{th} iteration for the update of the primal variable x_i. Accordingly, we introduce the variable β_i^k defined as:

$$\beta_i^k = \sum_{j\in N(i)\backslash i} -A_{i\to j}^T\lambda_{ij}^k. \tag{5.44}$$

Making use of the equation (5.15), by substituting λ_{ij}^k with $\lambda_{ij}^{k-1}+\rho(c_{ij}-A_{i\to j}x_i^{k-1}-A_{j\to i}x_j^{k-1})$ in equation (5.44) we have:

$$\begin{aligned}\beta_i^k &= \sum_{j\in N(i)\backslash i} -A_{i\to j}^T\{\lambda_{ij}^{k-1}+\rho(c_{ij}-A_{i\to j}x_i^{k-1}-A_{j\to i}x_j^{k-1})\}\\ &= \sum_{j\in N(i)\backslash i} -A_{i\to j}^T\lambda_{ij}^{k-1}+\rho\sum_{j\in N(i)\backslash i} -A_{i\to j}^T(c_{ij}-A_{i\to j}x_i^{k-1}-A_{j\to i}x_j^{k-1})\\ &= \beta_i^{k-1}+\rho\sum_{j\in N(i)\backslash i} -A_{i\to j}^T(c_{ij}-A_{i\to j}x_i^{k-1}-A_{j\to i}x_j^{k-1})\end{aligned} \tag{5.45}$$

Therefore, the updates in (5.14)-(5.15) reduce to the following updates:

$$x_i^{k+1} = \arg\min_{x_i}(f_i(x_i)+\beta_i^{k^T}x_i+\sum_{j\in N(i)\backslash i}\frac{\rho}{2}||c_{ij}-A_{i\to j}x_i-A_{j\to i}x_j^k||_2^2),\ i\in\nu \tag{5.46}$$

$$\beta_i^{k+1} = \beta_i^k+\rho\sum_{j\in N(i)\backslash i} -A_{i\to j}^T(c_{ij}-A_{i\to j}x_i^k-A_{j\to i}x_j^k),\ i\in\nu \tag{5.47}$$

By comparing (5.46)-(5.47) with (5.22)-(5.23) we see that using (5.22)-(5.23) require the need for calculating and sending a unique message $\lambda_{i|j}$ to each neighbour $j\in N(i)\backslash i$ in contrast to (5.46)-(5.47). The update in (5.47) has also the following properties:

Claim 1: By using (5.47) we are able to reduce the required memory at the node i if $n_i < |N(i)\backslash i|n_{ij}$, since $\beta_i\in\mathbb{R}^{n_i}$ and $\lambda_i\in\mathbb{R}^{|N(i)\backslash i|n_{ij}}$.

Claim 2: By using (5.47) we are able to reduce the number of transmitted messages by a factor $\bar{N}+1$, where $\bar{N}$ is the average degree (average number of neighbours of the nodes) of the network, as there is no need neither to send

different messages as $\lambda_{i|j}$, $j \in N(i)\backslash i$ nor to send β_i to any neighbour.
Claim 3: By using (5.47) we are able to reduce the required transmission bandwidth for data transmission by a further factor 2 as the messages are only comprised of the primal variables instead of the primal and dual variables.

It is good to mention that a simple synchronous update can also be derived from the synchronous AMM updates (5.22)-(5.23) as:

$$x_i^{k+1} = \arg\min_{x_i}(f_i(x_i) + \beta_i^{k^T} x_i + \sum_{j \in N(i)\backslash i} \frac{\rho}{2}||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j^k||_2^2), \ i \in \nu \tag{5.48}$$

$$\beta_i^{k+1} = \beta_i^{k-1} + \alpha \sum_{j \in N(i)\backslash i} -A_{i\to j}^T(2c_{ij} - 2A_{i\to j}x_i^k - A_{j\to i}x_j^{k-1} - A_{j\to i}x_j^{k+1}), \ i \in \nu. \tag{5.49}$$

Details of the derivation of the above updates can be found in Appendix B. (5.49) has almost the same properties as (5.47) but it requires additional storage to store β_i^{k-1} and x_j^{k-1}, $j \in N(i)\backslash i$. Therefore, it is recommended to use (5.22)-(5.23) in *asynchronous* systems while in *synchronous* systems the updates in (5.46)-(5.47) are recommended.

5.3.4 Consensus optimization via Async-AMM

The Async-AMM algorithm that is introduced in Section 5.3.1 can also be applied as an asynchronous distributed solver for the consensus optimization problems. Indeed, by substituting $A_{i\to j}$, $A_{j\to i}$ and c_{ij} with $\text{sign}(j-i)I_m$, $\text{sign}(i-j)I_m$ and 0 respectively, where I_m is an identity matrix of size m, the constrained convex problem (5.1), with general form of linear equalities, reduces to a consensus optimization problem of the form below:

$$\begin{aligned} &\min_{x_1,\ldots,x_{|\nu|}} \sum_{i\in\nu} f_i(x_i) \\ &\text{s.t.} \quad x_i = x_j \ \ \forall (i,j) \in \varepsilon. \end{aligned} \tag{5.50}$$

By making use of the aforementioned settings for $A_{i\to j}$, $A_{j\to i}$ and c_{ij}, the Async-AMM updates reduce to the following updates:

$$x_i^{k+1} = \arg\min_{x_i}\{f_i(x_i) + \sum_{j\in N(i)\backslash i}[-\text{sign}(j-i)\lambda_{j|i}^{k^T}x_i + \frac{\rho}{2}||x_i - x_j^k||_2^2]\},\ i\in\nu \tag{5.51}$$

$$\lambda_{i|j}^{k+1} = \arg\max_{\lambda_{i|j}}\{\text{sign}(j-i)\lambda_{i|j}^T(x_j^k - x_i^{k+1}) - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}^k||_2^2\},\ j\in N(i)\backslash i. \tag{5.52}$$

For the consensus problems of the form (5.50), the alternative synchronous updates (5.46)-(5.47) and (5.48)-(5.49) also reduce to the following updates (5.53)-(5.54) and (5.55)-(5.56) respectively:

$$x_i^{k+1} = \arg\min_{x_i}(f_i(x_i) + \beta_i^{k^T}x_i + \sum_{j\in N(i)\backslash i}\frac{\rho}{2}||x_i - x_j^k||_2^2),\ i\in\nu \tag{5.53}$$

$$\beta_i^{k+1} = \beta_i^k + \rho\sum_{j\in N(i)\backslash i}(x_i^k - x_j^k),\ i\in\nu \tag{5.54}$$

$$x_i^{k+1} = \arg\min_{x_i}(f_i(x_i) + \beta_i^{k^T}x_i + \sum_{j\in N(i)\backslash i}\frac{\rho}{2}||x_i - x_j^k||_2^2),\ i\in\nu \tag{5.55}$$

$$\beta_i^{k+1} = \beta_i^{k-1} + \alpha\sum_{j\in N(i)\backslash i}(2x_i^k - x_j^{k-1} - x_j^{k+1}),\ i\in\nu \tag{5.56}$$

By using the updates in (5.53)-(5.54) or in (5.55)-(5.56), there is no need to assign an identification (ID) number to the nodes of the network as a consequence of elimination of the sign function in comparison with (5.51)-(5.52). However, as was mentioned earlier the updates in (5.53)-(5.54) or in (5.55)-(5.56) can only be applied to *synchronous* systems while the updates in (5.51)-(5.52) can be applied to both *synchronous* and *asynchronous* systems.

5.4 Experimental Results

In order to evaluate the performance of the Async-AMM algorithm, we applied it to a consensus optimization problem over a graph, $G = (\nu, \varepsilon)$, of 25 nodes and

40 edges with grid connectivity pattern defined as below:

$$\begin{aligned} &\min_{w_1,\dots,w_{25}} \sum_{i=1}^{25} f_i(w_i) \\ &\text{s.t.} \quad w_i = w_j \;\; \forall (i,j) \in \varepsilon, \end{aligned} \tag{5.57}$$

where $f_i(w_i) = \frac{1}{2}||d_i - U_i w_i||_2^2$. We assumed that each node i has access to its own local data d_i and U_i and cooperatively try to estimate a parameter vector $w^o_{2\times 1}$ that satisfies the following equality:

$$d_{25\times 1} = U_{25\times 2} w^o_{2\times 1}, \tag{5.58}$$

where d_i and U_i are the i^{th} row of the vector d and matrix U respectively. By solving the problem (5.57) in a distributed processing manner, each node i is aimed to have an estimate on w^o via the variable w_i using its own local function $f_i(w_i)$.

In the experiments, by making use of (5.58), we generated the elements of the vector d by sampling the elements of the matrix U and the vector w^o from a Gaussian distribution with zero mean and unit variance. Fig. 5.2 illustrates the convergence rate of the Async-AMM algorithm in terms of the number of iterations required to reach to a target MSE$=10^{-10}$ for different values of the primal penalty parameter, ρ, (with the definition MSE $= \frac{1}{25}\sum_{i=1}^{25} ||w_i - w^o||_2^2$). The results in the Fig. 5.2 were obtained by setting $\alpha = \rho$ and activating

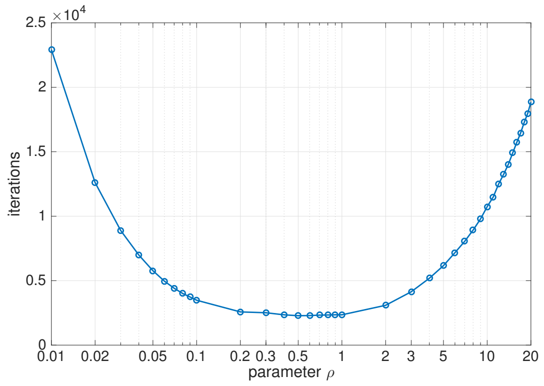


Figure 5.2: Illustration of the impact of the parameter ρ on the performance of Async-AMM.

one node per iteration, where at each iteration the node i =mod$(k, 25) + 1$ was activated. From the results in Fig. 5.2 we see that the optimal range of the parameter is $0.2 \leq \rho \leq 1$ and the fastest convergence rate is obtained by setting the primal penalty parameter ρ to 0.5. It is good to note that there is an alternative way of calculating the optimal value of the parameter ρ in which the optimal value is evaluated theoretically through an optimization process, see e.g. [56]. However, there is not a closed-form formula, for calculation of the optimal value of the penalty parameter, to be used for different problems with different objective functions. So for the sake of simplicity we evaluated the optimal value experimentally.

Fig. 5.3 shows the MSE of the estimated parameter vector for different dual penalty parameter α using the optimal primal penalty parameter $\rho = 0.5$. From the results in Fig. 5.3 it was observed that the Async-AMM algorithm has fastest convergence for $\alpha = 0.4$ and diverges for $\alpha > \rho$.

Fig. 5.4 illustrates the Normalised Mean Squared Error (NMSE) between the estimated parameter vector at the nodes of the network, at the target MSE=10^{-10}, as a measurement tool that measures the consensus between the nodes, where we used the definition NMSE=$\frac{(1/M)\sum_{(i,j)\in\varepsilon}||w_i - w_j||_2^2}{(1/2)\sum_{(i,j)\in\varepsilon}(||w_i||_2^2 + ||w_j||_2^2)}$. The results in the Fig. 5.4 were obtained by setting the primal penalty ρ to 0.5. By comparing the results in figures 5.3 and 5.4 we see that although Async-AMM has a slow convergence rate for small α, more consensus is obtained between the nodes of the network. The higher consensus between the nodes for small α is the consequence of re-

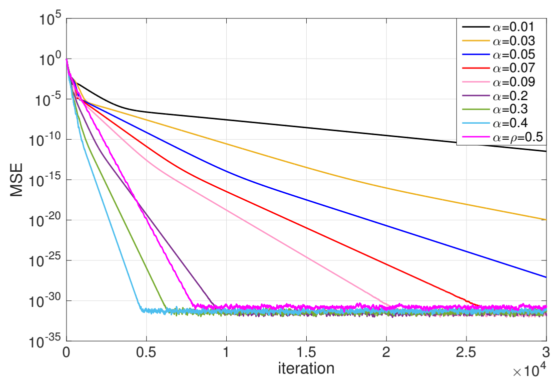


Figure 5.3: Illustration of the convergence behaviour of Async-AMM for different parameter α at the given $\rho = 0.5$.

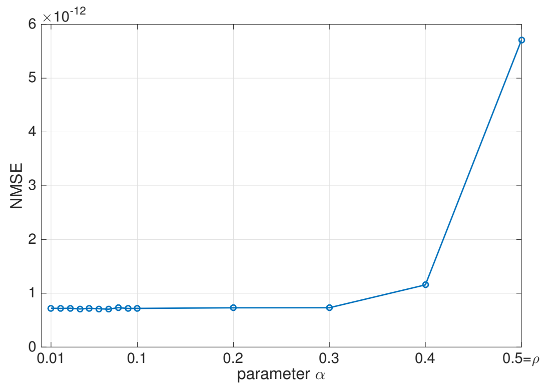
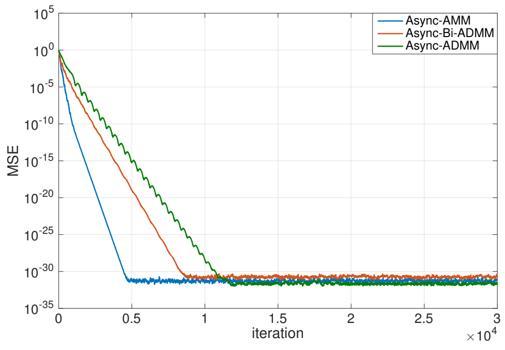


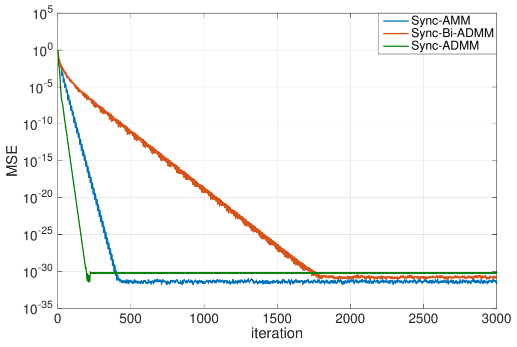
Figure 5.4: Illustration of the consistency of the agents in distributed consensus optimization in the form of NMSE across different α at the target MSE=10^{-10} using Async-AMM.

maining a lower residual for the equality constraints in (5.57) after convergence of the algorithm since α plays the role of learning rate, ρ, in the update of dual variable.

We compared the performance of the AMM algorithm with ADMM and Bi-ADMM algorithms and the results are plotted in Fig. 5.5. Fig. 5.5-a compares the algorithms in the asynchronous updating scheme while Fig. 5.5-b compares them in the synchronous updating scheme. For the synchronous case we used the alternative updates (5.55)-(5.56) and we referred to it as Sync-AMM. The results in Fig. 5.5 were obtained by setting ρ to 0.5 and α to 0.4 for AMM and setting ρ to 1 for ADMM since they were the optimal values for the algorithms. We

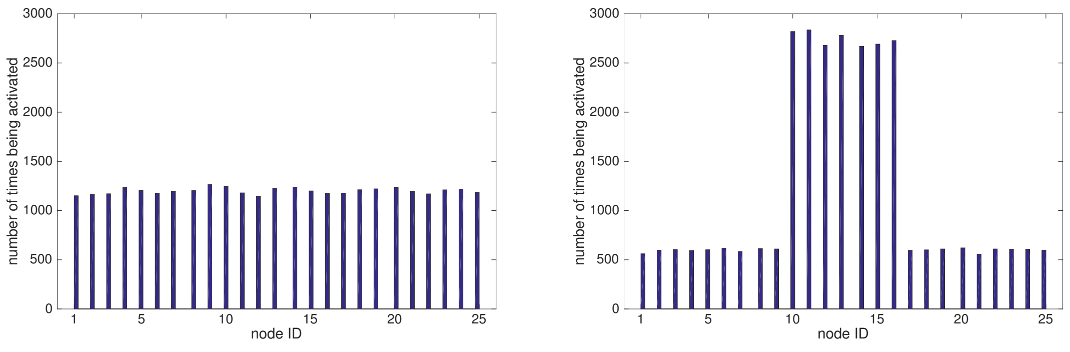


(a) Asynchronous



(b) Synchronous

Figure 5.5: Comparative performance of AMM, Bi-ADMM and ADMM.



(a) Experiment 1 (b) Experiment 2

Figure 5.6: Histogram of the sampled IDs.

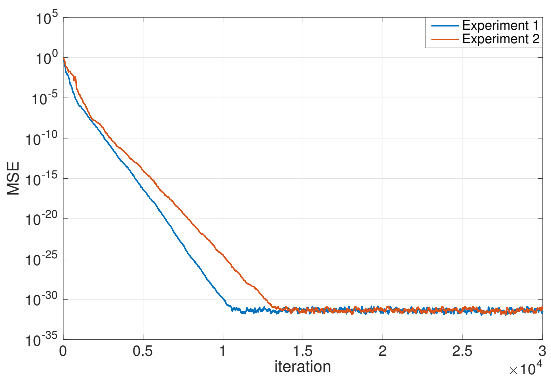


Figure 5.7: Illustration of the convergence behaviour of the network.

evaluated the optimality of $\rho = 1$ for ADMM through the same procedure that was done for extraction of the results in Fig. 5.2. For the case of Async-AMM and Async-Bi-ADMM we activated one node per iteration, where at each iteration the node $i = \text{mod}(k, 25) + 1$ was activated. For the case of Async-ADMM, at each iteration we activated one edge (one pair of nodes that are connected by an edge) of the graph along with their local master node, where at each iteration the edge $m = \text{mod}(k, 40) + 1$ was activated.

From the results in Fig. 5.5 we see that although a lower number of nodes participated per iteration using Async-AMM and Async-Bi-ADMM in comparison with Async-ADMM, they have a quicker rate of convergence. The results in Fig. 5.5-a imply that the auxiliary variable z that is used in Async-ADMM is not necessarily required to be used for convergence purposes and also imply the good

performance of Async-AMM. It also implies that applying Async-AMM results in lower communication cost since it requires lower number of iterations and consequently lower number of exchanged messages to reach to a certain level of accuracy. For the synchronous case, Sync-ADMM converges faster than Sync-AMM and Sync-Bi-ADMM since it exploits a global master (centralized) node in contrast to AMM and Bi-ADMM. From the results in Fig. 5.5 we also see different steady state response for each algorithm. This is based on the fact that although the algorithms converge with the rate of $O(1/k)$, the are other factors in the function $O(.)$ that affect both the learning rate as well as the steady state solutions. These factor generally depend on how the algorithms are derived and the parameters therein.
To evaluate the convergence rate of the proposed algorithm in terms of the amount of activation (participation) of the nodes, we did two experiments. In the two experiments we used different probability distributions for activation of the nodes. The probability distributions that we used include a uniform distribution along with a specific self-designed distribution that gives higher probability to a set of nodes compared to the others. To activate a node per iteration we sampled its identification number from the probability distributions. Fig. 5.6 displays the histogram of the sampled IDs, where we used 30000 iterations. Fig. 5.7 illustrates the convergence rate of Async-AMM for the two experiments where ρ and α were set to 0.5 and 0.4 respectively. From Fig. 5.7 we see that the convergence rate of the network is sensitive to the activation of the most passive node(s).

5.5 Summary

In this chapter, we introduced a fully asynchronous distributed algorithm for linearly constrained problems with more than one block of constraint. The algorithm was derived from a problem in which the auxiliary variable z was eliminated. As a consequence of eliminating the variable z, the proposed algorithm has lower computational complexity than the basic ADMM-based methods. By making use of the penalty method we were allowed to active one node per iteration similar to, but simpler than, Bi-ADMM. We also introduced efficient synchronous algorithms for the purpose of usage in synchronous systems. By tuning the parameter α in the proposed algorithm we were able to control the

consistency of the agents in the consensus application. The experimental results show that the proposed Async-AMM has faster convergence rate in comparison with the Async-ADMM [90] and Async-Bi-ADMM [202].
In the next chapter, we discuss distributed algorithms for blind source separation. The AMM algorithm that was proposed in this chapter forms a basis for distributed processing in Chapter 6.

6

Distributed processing of linear BSS over arbitrary graphs

6.1 Introduction

In this chapter, we implement distributed forms of the extended version of the adaptive BSS algorithm of [11, 109] over wireless sensor networks. We propose two distributed forms and refer to them as scheme 1 and scheme 2.

In scheme 1, each sensor node computes the estimate of one specific source signal based on the assumption that the parameter matrices of the BSS algorithm, the mixing or de-mixing matrices, have a similar sparsity structure as the Laplacian matrix of the graph of the network. Thus, a scalable algorithm can be obtained over graphs with pre-specified sparse connectivity patterns.

For scheme 1 algorithms that optimize the *de-mixing* matrix, the sources can be estimated at each node instantaneously by a weighted sum of the observed signals of the neighbour nodes. For scheme 1 algorithms that optimize the *mixing* matrix, the sources can be estimated in a distributed processing manner by converting the graph of the network into a probabilistic graphical model, without any changes in its connectivity pattern, and then using belief propagation algorithms [169,200,201] in an iterative manner to find the sources. After estimating the sources, in both mixing and de-mixing based BSS algorithms, the parameters can be updated at each node by communicating with the neighbour nodes. Applying the first scheme over the BSS algorithms that optimize the *de-mixing* matrix has the advantage of low communication cost and transmission power.

In scheme 2 each sensor node computes estimates of all source signals. It consists of two steps for each time sample. In the first step it solves a constrained convex optimization problem to estimate the sources given the current observations and the current estimate of the parameters. The objective function of the optimization problem forms a sum of N convex functions, over a graph $G = (\nu, \varepsilon)$ where ν denotes the set of nodes with cardinality $N = |\nu|$ and ε denotes the set of edges, each associated with an individual node of the network. By using the AMM algorithm, which was discussed in the Chapter 5, a distributed solution to the constrained problem can be obtained. In the second step, the parameters are updated given the newly estimated sources. The second step is local and requires no communication across the nodes. The proposed distributed processing strategy enables the algorithm to be implemented on connected graphs of any connectivity pattern without significant degradation of the performance of the BSS algorithm over that of a corresponding centralized algorithm. The second scheme is particularly computationally efficient if there is a low number of sources while there are higher number of sensor nodes.

When our distributed BSS algorithms (scheme 1 & 2) are applied to a properly designed sparse network, the energy requirement for data transmission is low. An appropriate selection of the connections means that no transmission over long distances is required.

In publication I [5], we applied the scheme 2 on BSS algorithms that optimize the *mixing* matrices. In the rest of this chapter we will see that applying the scheme 2 on BSS algorithms that optimize the *de-mixing* matrices requires lower transmission power compared to the *mixing* matrix based BSS algorithms. As was mentioned earlier the latter privilege is also seen in the scheme 1 since applying the *mixing* matrix based BSS algorithms in the form of scheme 1 require iterative procedure for source estimation in comparison with the *de-mixing* matrix based approaches. Finally we will see that for the *de-mixing* matrix based BSS algorithms, the minimum transmission power and higher scalability is achieved

when the scheme 1 is applied in comparison with the scheme 2.
We ascertained the performance of the proposed distributed BSS with experiments for the case of delay-restricted environments where the observations are obtained from the linear instantaneous mixture of the sources. The method is easily extended to apply to convolutional scenarios using conventional frequency domain approaches.
The remainder of this chapter is organized as follows. In Section 6.2 we formulate the linear BSS problem and introduce a mixing and a de-mixing matrix based centralized processing solution for the problem. In Section 6.3, we propose distributed processing solution for the linear BSS algorithms. We discuss the experimental results in Section 6.4 followed by a conclusion in Section 6.5.

6.2 Centralized blind source separation

In this section we formulate the linear BSS problem and review two centralized algorithms [11, 109] that can be used as a solution for the problem.
As was mentioned in Chapter 2, a linear model can be used to describe N-dimensional observations that are the result of an instantaneous mixing of M sources and additive noise:

$$\mathbf{b}_{N\times 1}(t) = A_{N\times M}\mathbf{s}_{M\times 1}(t) + \mathbf{n}_{N\times 1}(t), \qquad t = 1, ..., T \tag{6.1}$$

where we provided dimensionalities as subscripts, and where $\mathbf{s}$, $\mathbf{b}$ and $\mathbf{n}$ denotes the source, observed and noise data respectively and A is the mixing matrix. We denote by $s(t)$, $b(t)$ and $n(t)$ the samples of the random vectors $\mathbf{s}(t)$, $\mathbf{b}(t)$ and $\mathbf{n}(t)$ at time index t respectively.
In BSS, it is assumed that only data observations, $b(t)$, are available while there is no information on the matrix A, $s(t)$ and $n(t)$. The objective is to estimate $s(t)$ for $t = 1, ..., T$. Many algorithms have been proposed such as [11, 12, 57, 86, 109], as a solution for linear BSS. Their differences are in the prior assumption about the size of N and M, the existence of noise in the model and the objective or cost function that they use for optimization purposes.
In the following two subsections we briefly discuss two centralized BSS algorithms that enjoy the equivariant property, that is, the performance of separation is

independent of the hardness of the mixture for example when the mixing matrix is ill-conditioned [11, 30, 36].

6.2.1 Maximum-Likelihood Approach

In this section we discuss an adaptive BSS algorithm based on the optimization of *mixing* matrix. To extract a set of signals from their mixture, requires a set of conditions to be satisfied. A natural and sufficient set of conditions is time-invariance of the mixing and source signals that are both stationary and mutually independent. Following [113] we also assume that the signals are white,

$$\mathrm{p}(\mathbf{b}(1), ..., \mathbf{b}(T)|A) = \prod_{t=1}^{T} \mathrm{p}(\mathbf{b}(t)|A). \tag{6.2}$$

As we assume no time dependencies we will omit the time argument where that causes no confusion.
As objective function we use the likelihood function $\mathrm{p}(b(t)|A)$. Using the law of total probability, it can be expressed as:

$$\mathrm{p}(b|A) = \int \mathrm{p}(b|\mathbf{s}, A)\mathrm{p}(\mathbf{s})\mathrm{d}\mathbf{s}. \tag{6.3}$$

By assuming a Gaussian additive noise, the conditional probability $\mathrm{p}(b|\mathbf{s}, A)$ can be modelled with a Gaussian distribution.
To satisfy the independence assumption over the sources, a fully factorizable prior distribution $\mathrm{p}(\mathbf{s})$ should be used in (6.3). For over-complete (under-determined) representations ($N < M$), there is not a unique expression for the observations. Therefore, it is natural to look for priors that assume sparse representations, e.g. priors with high kurtosis such as Laplacian, since only N non-zero values out of M values are required to represent the observations [113]. As [109] is basically developed for over-complete representations, a (fully factorizable) *zero* mean Laplacian prior with *diagonal* covariance matrix is used for the distribution of the sources. The Laplacianity of the prior itself makes the integral in (6.3) intractable. Therefore in [109] an approximation of the logarithm of (6.3) is used as the objective function. To derive a learning rule, one can use a gradient ascent algorithm to maximize $\log \mathrm{p}(b|A)$. In [109] a stochastic natural gradient [8] ascent

is used to stabilize the learning algorithm:

$$\triangle A \propto AA^T \frac{\partial}{\partial A} \log \mathrm{p}(b|A) \approx -AF[y], \tag{6.4}$$

which by using the stochastic learning rule leads to the following update:

$$A(t+1) = A(t) - \mu(t)A(t)F[y(t)], \tag{6.5}$$

where $F[y(t)] = I - \psi[y(t)]y(t)^T$, $y(t)$ is an estimate of $s(t)$, I is the identity matrix, $\mu(t) > 0$ is a variable learning rate and ψ is the marginal score function defined as:

$$\begin{aligned} \psi[\mathbf{y}] &= [\psi_1[\mathbf{y}_1], ..., \psi_M[\mathbf{y}_M]]^T \\ \psi_i[\mathbf{y}_i] &= -\frac{d}{d\mathbf{y}_i} \ln \mathrm{p}(\mathbf{y}_i). \end{aligned} \tag{6.6}$$

For Laplacian priors, $\psi_i[\mathbf{y}_i]$ can be approximated as $\mathrm{sign}[\mathbf{y}_i]$.

According to [109, 113] for over-complete (under-determined) representations $(N < M)$, $y(t)$ is obtained by a probabilistic formulation as:

$$\begin{aligned} y(t) =& \arg\max_s \mathrm{p}(s|b(t), A(t)) \\ =& \arg\max_s \mathrm{p}(b(t)|A(t), s)\mathrm{p}(s). \end{aligned} \tag{6.7}$$

However in this chapter we assume that the number of observations (sensor nodes) is greater than or equal to the number of sources $(N \geq M)$. For the case of low noise and under-complete (over-determined) representations $(N \geq M)$, where A is full column rank, we are able to relax the prior sparsity assumption over the source signals and can estimate the sources at time index t via the following transformation:

$$y(t) = A^{\dagger}(t)b(t) \tag{6.8}$$

where $A^{\dagger}(t)$ is the pseudo inverse of the approximation of A at time index t. However, in the over-determined representations $(N \geq M)$, we can still preserve the sparsity assumption about the source signals, by using a *zero* mean Laplacian prior and considering the sign function as the marginal score function, when estimating the mixing matrix. The motivation for considering the zero mean Laplacian distribution, as a distribution with high kurtosis, is to be able to capture the underlying distribution of human speech signals.

As no information is available about the energy of the source signals, an arbitrary diagonal matrix is used as a prior assumption over the covariance matrix of the sources. The latter assumption forces the columns of the matrix A to be updated such that the variance of the corresponding estimated source signal to be equal to the arbitrary fixed value over time. This results in numerical instability over the elements of mixing matrix A in the time periods in which at least some of the sources are silent. A scaled version of the columns of the mixing matrix $A\Omega$, where Ω is a diagonal scaling matrix containing large factors, will be learned to compensate the silence periods of the sources. According to [10] the trajectory of learning will be orthogonal to the set of all possible mixing matrices that are only different in terms of their column vectors' norm if $F[y(t)]$ is chosen as follows:

$$F[y(t)] = \Lambda(t) - \psi[y(t)]y(t)^T \tag{6.9}$$

where $\Lambda(t) = \mathrm{diag}[\mathrm{diag}[\psi[y(t)]y(t)^T]]$. It should be noted that diag of a matrix returns the diagonal elements of that matrix in a vector form and diag of a vector creates a diagonal matrix whose diagonal elements are defined in the vector. This means that the diagonal elements of $F[y(t)]$ are zero. [10] is proposed to be used in the de-mixing matrix based BSS algorithms while with a similar methodology we applied it to the mixing matrix based approaches. The proof of the correctness of $\Lambda(t)$ for the mixing matrix based BSS algorithm can be found in Appendix C. The main purpose of the equation (6.9) is to preserve the stability of the BSS algorithm when the number of sources is overestimated.

Algorithm 8 summarizes our target BSS algorithm based on maximum likelihood where $N \geq M$. Applying the Algorithm 8 in cases where the real number of original sources, M, is overestimated by $P > M$, leads to estimating M original source signals together with $P - M$ zero signals if the noise level in the linear model (6.1) is zero. The aforementioned outcome is the consequence of using $\Lambda(.)$, as the stability controller of the BSS algorithm, along with using an appropriate marginal score function, $\psi[.]$, that can capture the underlying distribution of M original source signals as well as the zero signals as signals with high kurtosis.

Algorithm 8 An on-line centralized linear BSS Algorithm based on optimization of *mixing* matrix for a time sequence of T time samples.

```
Randomly initialize $A$
**for** $t = 1, ..., T$ **do**
    $y(t) = A^{\dagger}(t)b(t)$
    $A(t+1) = A(t) - \mu(t)A(t)(\Lambda(t) - \psi[y(t)]y(t)^T)$
**end for**
```

6.2.2 Minimum Average Mutual Information Approach

In this section we discuss an adaptive BSS algorithm based on the optimization of *de-mixing* matrix. As was mentioned in the previous section, to extract a set of signals from their mixture a set of conditions need to be satisfied. One of the crucial conditions, particularly for statistically independent source signals, is the prior independency assumption about the source signals. In a mathematical framework, the dependency between a set of random variables, $\mathbf{y}_1, ..., \mathbf{y}_M$, can be measured by the Kullback-Leibler divergence [40] between the joint and marginal distributions of the random variables and is defined as:

$$K(W) = \int \mathrm{p}(\mathbf{y})\mathrm{log}\frac{\mathrm{p}(\mathbf{y})}{\prod_{i=1}^{M}\mathrm{p}_i(\mathbf{y}_i)} \tag{6.10}$$

where $\mathbf{y} = [\mathbf{y}_1, ..., \mathbf{y}_M]^T$ and W is the transformation matrix that maps the input data $\mathbf{b}$ to the output data $\mathbf{y}$ as $\mathbf{y} = W\mathbf{b}$. The Kullback-Leibler divergence of (6.10) is sometimes referred to as the average mutual information [11]

$$K(W) = -H(\mathbf{y}) + \sum_{i=1}^{M} H(\mathbf{y}_i) \tag{6.11}$$

where $H(.)$ is the differential entropy.

Minimization of the average mutual information (6.11) leads to separation of the sources. To develop an on-line learning rule, one can use a gradient descent algorithm. In [11] the natural gradient descent is used to increase the stability of the learning algorithm as:

$$\triangle W \propto \frac{\partial K(W)}{\partial W} W^T W, \tag{6.12}$$

Algorithm 9 An on-line centralized linear BSS Algorithm based on optimization of *de-mixing* matrix for a time sequence of T time samples.

```
Randomly initialize $W$
**for** $t = 1, ..., T$ **do**
    $y(t) = W(t)b(t)$
    $W(t+1) = W(t) + \mu(t)(\acute{\Lambda}(t) - \varphi[y(t)]y(t)^T)W(t)$
**end for**
```

which implies

$$W(t+1) = W(t) + \mu(t)\acute{F}[y(t)]W(t), \tag{6.13}$$

where $\acute{F}[y(t)] = I - \varphi[y(t)]y(t)^T$ and $y(t) = W(t)b(t)$.

In [11] the entropy is approximated using a truncated Gram-Charlier expansion. The latter approximation results to a component-wise non-linear polynomial function (known as activation function from neural network point of view) $\varphi[y(t)]$ of degree 11. The correct choice of $\varphi[y(t)]$ depends on the distribution of the original sources. The hyperbolic tangent is a widely used [2, 20, 108] activation function which functions well for super Gaussian distributions, distributions with high kurtosis, such as Laplacian distribution. Therefore we substitute the polynomial function with the hyperbolic tangent:

$$\begin{aligned} \varphi[\mathbf{y}] &= [\varphi_1[\mathbf{y}_1], ..., \varphi_M[\mathbf{y}_M]]^T \\ \varphi_i[\mathbf{y}_i] &= \vartheta \tanh(\theta \mathbf{y}_i), \end{aligned} \tag{6.14}$$

where ϑ and θ are positive constants. In fact, $\vartheta\tanh(\theta\mathbf{y}_i)$ can be interpreted as the marginal score function derived from a smooth approximation of the Laplacian distribution as:

$$\vartheta\tanh(\theta\mathbf{y}_i) = -\frac{d}{d\mathbf{y}_i}\ln \hat{\mathrm{p}}(\mathbf{y}_i) \tag{6.15}$$

where $\hat{\mathrm{p}}(\mathbf{y}_i)$ is the approximate zero mean Laplacian distribution defined as:

$$\hat{\mathrm{p}}(\mathbf{y}_i) = \frac{1}{\zeta}\cosh^{-\vartheta/\theta}(\theta\mathbf{y}_i) \tag{6.16}$$

where ζ is a normalizing coefficient.

As was mentioned earlier the motivation for considering the Laplacian distribution is to make the algorithm suitable for separation of human speech signals.

According to [10], the adaptive BSS algorithm based on minimization of the aver-

age mutual information can also be extended to be more stable in situations where the number of sources is overestimated by using $\acute{F}[y(t)] = \acute{\Lambda}(t) - \varphi[y(t)]y(t)^T$ where $\acute{\Lambda}(t) = \text{diag}[\text{diag}[\varphi[y(t)]y(t)^T]]$.
It is important to note that the adaptive learning rule (6.13) is proposed to be used in noiseless determined cases ($N = M, n(t) = 0 \ \forall t$). However, the applicability of (6.13) for over-determined ($N > M$) noisy observations has been proved in [205].
Algorithm 9 summarizes our target BSS algorithm based on minimum average mutual information where $N \geq M$. Corresponding to the mixing matrix based BSS Algorithm 8, applying the Algorithm 9 in cases where the real number of original sources, M, is overestimated by $P > M$, leads to estimating M original source signals together with $P - M$ zero signals if the noise level in the linear model (6.1) is zero.

6.3 Distributed Processing Approach

In this section we discuss distributed solutions to the BSS problem. We propose two distributed schemes and we denote them as scheme 1 and scheme 2. In scheme 1 each node estimates one specific source signal while in scheme 2 each node estimates the entire source signals through a consensus optimization process. Both schemes enjoy a fully shared computation process, which is achieved by partitioning the source separation algorithm and allocating partial modules to each node of the network. In Section 6.3.1 we apply the BSS algorithm over graphs in the form of scheme 1. In Section 6.3.2 we develop distributed BSS algorithms over graphs in the form of scheme 2.

6.3.1 Scheme 1

In this section we first explain implementation of scheme 1 on the de-mixing matrix based BSS Algorithm 9 followed by an explanation on implementation of scheme 1 on the mixing matrix based BSS Algorithm 8. In the sequel, we implement the de-mixing matrix based BSS Algorithm 9 on *arbitrary* graphs in the form of scheme 1 followed by introducing a simpler algorithm along with exploiting an l_1 norm regularization for parameters' update.

De-mixing matrix based approach

To develop distributed algorithms for blind source separation, one can partition the source estimation and the parameter update steps, in the Algorithms 8 and 9, and allocate each of them to a sensor node on a network of nodes. In this fully shared computation strategy, each sensor node would be responsible for tracking a specific source signal while participating in optimization of a partition of the mixing or de-mixing matrix. To see how one can develop the aforementioned distributed learning strategy, let us consider a fully connected bi-directional graph of N nodes in which the edge between the node i and j is weighted by the $(ij)^{th}$ and $(ji)^{th}$ element of the parameter matrix (W or A). Making use of the *de-mixing* matrix based BSS Algorithm 9, we can allocate the task of estimation of the i^{th} source signal (the i^{th} row of $y(t)$) to the i^{th} node of the graph as a weighted sum of the observation of the neighbour nodes:

$$y_i(t) = \sum_{k=1}^{N} W_{ik} b_k(t), \tag{6.17}$$

where only the i^{th} row of the de-mixing matrix, W, is used. To update the de-mixing matrix, we can also allocate the update of the i^{th} column of W to the i^{th} node of the graph as:

$$W_{mi}(t+1) = W_{mi}(t) + \mu(t)\acute{F}[y(t)]_m W^i(t), \qquad m = 1, ..., N \tag{6.18}$$

where $\acute{F}[y(t)]_m$ and $W^i(t)$ denotes the m^{th} row and i^{th} column of $\acute{F}[y(t)]$ and $W(t)$ respectively. Using a column-wise update, the update of each element of the column vector requires an observation only on the last estimate of that column as can be verified in (6.18). Applying (6.17)-(6.18) on a network of nodes has exactly the same performance, e.g. signal to interference ratio, as the centralized Algorithm 9 if the graph of the network is *fully* connected. In the aforementioned fully connected network, M (the number of original sources) nodes will track M source signals while the others will estimate the zero signals if the noise level is zero. Using the aforementioned distributed processing technique, the entire network enjoy a fully shared computation. We refer to this approach as scheme 1.

Mixing matrix based approach

We are also able to apply the scheme 1 on the *mixing* matrix based BSS Algorithm 8. However, in order to estimate the i^{th} source signal at node i via Algorithm 8, there is the need for calculation of the i^{th} row of the inverse matrix of A. Direct calculation of the inverse of a matrix eliminates the ability for a fully shared computation and suggests a centralized processing unit. To eliminate the need for direct calculation of the inverse matrix, we can follow the iterative algorithms such as GaBP [169] or LiCD [201], that was discussed in Chapter 3, to estimate the i^{th} source at node i using only the i^{th} column of the mixing matrix A. After estimating the source signals on a fully connected graph, to update the mixing matrix we can allocate the update of the i^{th} row of A to the i^{th} node of the graph as:

$$A_{im}(t+1) = A_{im}(t) - \mu(t)A_i(t)F[y(t)]^m, \qquad m = 1, ..., N \tag{6.19}$$

where $F[y(t)]^m$ and $A_i(t)$ denotes the m^{th} column and i^{th} row of $F[y(t)]$ and $A(t)$ respectively. Using a row-wise update, the update of each element of the row vector requires an observation only on the last estimate of that row as can be verified in (6.19). With the assumption that sufficient number of iterations are used for the convergence of the message passing algorithms [169, 201], applying the mixing matrix based BSS Algorithm 8 in the form of scheme 1 on a fully connected network has similar performance to its corresponding centralized algorithm.

It is important to note that applying the *mixing* matrix based BSS Algorithm 8 in the form of scheme 1 requires tens of thousands of iterations, specially for general matrices (matrices that are not necessarily symmetric walk-summable) [201], to estimate the source signals. This is in contrast to the *de-mixing* matrix based BSS Algorithm 9 in which the sources are obtained as a weighted sum of the observations of the nodes, without the need for iterations, as shown in (6.17). Since in each iteration of the message passing algorithms there is the need for communication between the nodes, the total power consumption of the transmission at each node increases linearly, proportional to the number of iterations, as:

$$P_{Tx,\text{Total}} = kP_{Tx}, \tag{6.20}$$

where P_{Tx} and k denote the power consumption of the transmission at each node and the total number of iterations respectively. Because of the aforementioned issue and to conserve energy on a WSN we do not pursue implementing the *mixing* matrix based BSS Algorithm 8 in the form of scheme 1 in this thesis.

De-mixing matrix based BSS on arbitrary graphs

Not only the distributed processing of mixing matrix based BSS Algorithm 8 in the form of scheme 1 would be a burden, but also distributed processing of the *de-mixing* matrix based BSS Algorithm 9 in the form of scheme 1 on a *fully* connected graph may conflict with the scalable nature of the distributed processing approaches. It may even result in a high power consumption of the transmission on large-scale problems since there is the need for transmission to long distances. To overcome the aforementioned issue one may think of applying the scheme 1 on a properly designed sparse network. A properly designed network is a network in which there is no connection between the nodes with long distances.

Applying the *de-mixing* matrix based BSS Algorithm 9 in the form of scheme 1 on a sparse static graph $G = (\nu, \varepsilon)$ with $|\nu| = N$ imposes the following constraint over the domain of the objective function (here the average mutual information function) that is used for BSS:

$$\begin{aligned} &W \in C_1, \\ &C_1 = \{W \in \mathbb{R}^{N\times N} | W_{ij} = 0 \text{ if } (i,j) \notin \varepsilon\}. \end{aligned} \tag{6.21}$$

Any de-mixing matrix W in the set C_1 will have the same sparsity structure as the Laplacian matrix of the graph. The Laplacian matrix Q of a graph $G = (\nu, \varepsilon)$ is a matrix whose elements are evaluated as follows:

$$Q_{ij} = \begin{cases} -1 & (i,j) \in \varepsilon \\ \deg\{\nu_i\} & i = j \\ 0 & \text{otherwise} \end{cases} \tag{6.22}$$

where $\deg\{\nu_i\}$ denotes the degree of the node i.

Fig. 6.1 shows a sample graphical representation of the scheme 1. Given the constraint in (6.21), we can formulate a constrained optimization problem for

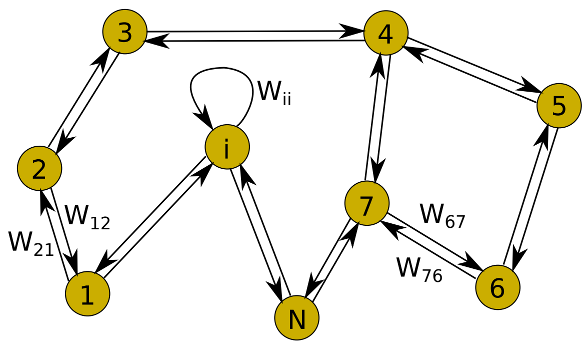


Figure 6.1: Illustration of a sample graph of scheme 1. The neighbour of node i includes $\{1, i, N\}$.

distributed BSS on an arbitrary graph as:

$$\begin{aligned} &\min_{W} \quad K(W) \\ &\text{s.t.} \quad W \in C_1 \end{aligned} \tag{6.23}$$

To address the constrained problem in (6.23) one can use the projected gradient descent method [61, 112]. Minimization of the objective function, $K(W)$, on a feasible set C_1 via the projected gradient descent method is defined by:

$$W(t+1) = \Pi_{C_1}(W(t) - \mu(t)\nabla K(W)) \tag{6.24}$$

where Π_{C_1} denotes the projection into feasible set C_1. There are other variants of projected gradient method, e.g. projected stochastic gradient descent method, where a stochastic gradient descent is used for minimization of the objective function.

Making use of the stochastic natural gradient descent learning rule, used in the Algorithm 9 for minimization of the objective function (average mutual information), projection into the set C_1 suggests the following updates at the i^{th} node of an arbitrary graph $G = (\nu, \varepsilon)$:

$$y_i(t) = \sum_{k \in N(i)} W_{ik}(t) b_k(t), \tag{6.25}$$

$$W_{mi}(t+1) = W_{mi}(t) - \mu(t) \sum_{k \in N(i) \setminus m} \varphi_m[y_m(t)] y_k(t) W_{ki}, \quad \forall m \in N(i), \tag{6.26}$$

where $N(i)$ denotes the neighbours of node i include i itself and $N(i)\setminus m$ excludes node m from the neighbours of node i.

Algorithm 10 shows the distributed BSS algorithm based on the minimum average mutual information in form of scheme 1 on an arbitrary graph.

Pairwise mutual information minimization

It is good to compare (6.25)-(6.26) with (6.17)-(6.18). In our comparison we assume that the number of sources is equal to the number of observations ($N = M$). Given the aforementioned assumption, it can be easily verified that in (6.18) each element, W_{ij}, of the i^{th} row of de-mixing matrix is evaluated as:

$$W_{ij} = \arg\min_{W_{ij}} \sum_{k=1,k\neq i}^{N} I(\mathbf{y}_i, \mathbf{y}_k), \qquad j = 1, ..., N \tag{6.27}$$

where $I(\mathbf{y}_i, \mathbf{y}_j)$ is the mutual information between $\mathbf{y}_i$ and $\mathbf{y}_j$ defined as:

$$\begin{aligned} I(\mathbf{y}_i, \mathbf{y}_j) &= H(\mathbf{y}_i) + H(\mathbf{y}_j) - H(\mathbf{y}_i, \mathbf{y}_j) \\ &= H(\mathbf{y}_i) - H(\mathbf{y}_i|\mathbf{y}_j) \\ &= H(\mathbf{y}_j) - H(\mathbf{y}_j|\mathbf{y}_i) \end{aligned} \tag{6.28}$$

From (6.27) we see that the i^{th} row of the de-mixing matrix aims to learn a direction such that the projection of the observed data in that direction extracts a source signal, say the i^{th} source signal, that is independent to $(N-1)$ other source signals. The aforementioned projection is achieved via (6.17), where N observations are used. In fact N source signals contribute in learning the appropriate direction of the projection while N observations are used in such a projection. This equality between the number of observations and the total number of sources makes the BSS problem to be determined. The (over) determinacy is a necessary condition in the standard independent component analysis (ICA) algorithms as otherwise the decomposition is not guaranteed to be unique from the centralized point of view.

In contrast to (6.18), in (6.26) each non-zero element, $W_{ij}, (i, j) \in \varepsilon$, of the i^{th} row of the de-mixing matrix is evaluated as:

$$W_{ij} = \arg\min_{W_{ij}} \sum_{k\in N(j)\backslash i} I(\mathbf{y}_i, \mathbf{y}_k), \qquad \forall j \in N(i) \tag{6.29}$$

From the (6.29) we see that the i^{th} row of the sparse de-mixing matrix is aimed to learn a direction such that the projection of the observed data in that direction

Algorithm 10 An on-line de-centralized linear BSS Algorithm based on minimization of average mutual information (Scheme1-MAMI) for a time sequence of T time samples.

```
Initialize $W_{ii}(1) = 1, W_{ij}(1) = W_{ji}(1) = 0$ for all $i \in \nu, (i,j) \in \varepsilon$
for $t = 1, ..., T$ do
    Estimate $y_i(t)$ at node $i$ using (6.25).
    Update $W_{mi}(t), m \in N(i)$ at node $i$ using (6.26).
end for
```

extracts a source signal independent to $T_S - 1$ other source signals, where T_S is calculated as:

$$T_S = |\bigcup_{k \in N(i)} N(k)|, \tag{6.30}$$

and $\bigcup$ denotes the union operator. This is when only $|N(i)|$ observations are used in such a projection, see (6.25). The aforementioned inconsistency between the number of contributed sources, T_S, and the number of used observations, $|N(i)|$, makes the BSS problem in the i^{th} node somehow under-determined, since $|N(i)| \leq T_S$. Therefore, we expect a significant degradation of the performance of the BSS Algorithm 9 when is applied in the form of scheme 1 over graphs with high degree of sparsity.

In order to resolve the under-determinacy problem we suggest to evaluate W_{ij} only as a minimizer of $I(\mathbf{y}_i, \mathbf{y}_j)$ instead of the minimizer of the average mutual information $\sum_{k \in N(j) \backslash i} I(\mathbf{y}_i, \mathbf{y}_k)$ as below:

$$W_{ij} = \arg\min_{W_{ij}} I(\mathbf{y}_i, \mathbf{y}_j), \qquad \forall j \in N(i) \backslash i \tag{6.31}$$

where we excluded the $W_{ii}, \forall i \in \nu$ from learning.

Using (6.31) reduces T_S to $|N(i)|$ and makes the BSS problem at the i^{th} node somewhat determined. However, the real indeterminacy at the i^{th} node depends on the real number of original independent sources that contribute in the local observations b_j, $j \in N(i)$. Therefore the indeterminacy completely vanishes if the following condition is satisfied:

Condition 1: Given the learning rule (6.31), to be able to estimate one source signal at the node i it is necessary that $|N(i)|$, the number of local observations, to be greater than or equal to the total number of independent sources that contribute in the local observations b_j, $j \in N(i)$.

Making use of the stochastic natural gradient learning rule, evaluation of W_{ij} via (6.31) suggests the following update:

$$W_{ij}(t+1) = W_{ij}(t) - \mu(t)\varphi_i[y_i(t)]y_j(t)W_{jj}, \quad i \neq j. \tag{6.32}$$

Since in the aforementioned learning strategy, each element of the de-mixing matrix is evaluated as a minimizer of only the mutual information between a pair of nodes, we refer to it as the minimum *pairwise* mutual information approach. It is good to note that the distributed implementation of the minimum *pairwise* mutual information based BSS over arbitrary graphs in the form of scheme 1 enables tracking of a specific source signal in the nodes that are at least two communication hops apart. Algorithm 11 shows the distributed BSS algorithm based on the minimum pairwise mutual information in form of scheme 1 on an arbitrary graph.

Making use of the distributed BSS Algorithm 11 over a graph with the assumption that the condition 1 is met, the estimated sources at an arbitrary subset of nodes would be mutually independent if the sub-graph of the subset nodes is *fully* connected. The *full* connectivity of the sub-graph implies a pairwise independence over all pairs of estimated sources in the subset nodes which in turn implies mutual independence according to [37, Theorem 11].

Since pairwise independence implies the mutual independence in linear instantaneous ICA [37, Theorem 11], implementation of the minimum pairwise mutual information Algorithm 11 on a *fully* connected graph, or in a centralized processing manner, leads to minimization of $K(W)$ in (6.10) similar to the centralized Algorithm 9.

The centralized update of the de-mixing matrix for minimization of the pairwise mutual information is defined as:

$$W(t+1) = W(t) + \mu(t)\acute{F}[y(t)]\mathrm{diag}[\mathrm{diag}[W(t)]]. \tag{6.33}$$

where $\acute{F}[y(t)] = \acute{\Lambda}(t) - \varphi[y(t)]y(t)^T$. By initializing the de-mixing matrix with the identity matrix, (6.33) reduces to:

$$W(t+1) = W(t) + \mu(t)\acute{F}[y(t)]. \tag{6.34}$$

By comparing (6.34) with (6.13) we see a reduction of computational complexity in (6.34) compared to (6.13). However, by using (6.34) instead of (6.13)

Algorithm 11 An on-line de-centralized linear BSS Algorithm based on minimization of pairwise mutual information (Scheme1-MPMI) for a time sequence of T time samples.

```
Initialize $W_{ii}(1) = 1, W_{ij}(1) = W_{ji}(1) = 0$ for all $i \in \nu, (i, j) \in \varepsilon$
for $t = 1, ..., T$ do
    for all $i \in \nu$ do
        $y_i(t) = \sum_{k \in N(i)} W_{ik}(t) b_k(t)$.
        $W_{mi}(t+1) = W_{mi}(t) - \mu(t)\varphi_m[y_m(t)]y_i(t)W_{ii}(t), \quad m \neq i$.
        $W_{ii}(t+1) = W_{ii}(t)$.
    end for
end for
```

we may lose the equivariant property since the serial update (6.13) is replaced with the parallel update (6.34) [30]. It is good to note that the simple parallel update (6.34) has been already investigated for *centralized* applications in some literatures, e.g. [130, 173].

l_1 norm regularization

According to the condition 1, in order to achieve a successful separation at the i^{th} node of a network via Algorithm 11, the number of neighbours of the i^{th} node should be greater than or equal to the number of local original sources that are captured by its neighbours. Since there may not be an exact prior knowledge about the number of original sources, we are motivated to design graphs with high degree of vertices. This can be achieved by increasing the distance threshold of friendship between the nodes, where each node is connected to another node if their distance is less than the distance threshold.

Due to the physical nature of the sound, increasing the distance threshold of friendship between the nodes can reduce the correlation between the observations of the nodes that are two communication hops apart. This itself enables a node to use the observations of a high number of relevant nodes for tracking a source signal at a particular location. However, increasing the distance threshold of friendship between the nodes results in communication between the nodes that are relatively far which consequently increases the required transmission power, channel congestion and interference. This increment in transmission power will be redundant when the number of local original sources around a node is sufficiently low.

To tackle the aforementioned problem, one can think of eliminating the need for transmission of non-informative messages between the nodes, e.g. messages that contain zero values, to reduce the required transmission power. Since the message that is transmitted from node i to node j at time index t is comprised of $W_{ji}(t)b_i(t)$ and $y_i(t)$, we consider it as a zero valued message if both $W_{ji}(t)b_i(t)$ and $y_i(t)$ are zero.

We consider the following condition:

Condition 2: The noise level in the linear model (6.1) is zero. This condition provides the opportunity of tracking either original signals or zero signals without the possibility of tracking noise signals.

With the assumption that the condition 2 is met, to investigate in which case we can have or produce zero valued messages, let us consider a fully connected graph $G = (\nu, \varepsilon)$ with $|\nu| = N$. We assume that there are $M < N$ original sources that need to be tracked via the nodes of the graph G. As was mentioned earlier, in the aforementioned graph M nodes will track M original source signals while $N - M$ nodes will track the zero signals. Accordingly, we define two sets, Θ_S and Θ_Z, where Θ_S and Θ_Z denotes the set of nodes that track the source signals and zero signals respectively. Since all the messages that are transmitted from the node with ID $\in \Theta_S$ to its neighbours include the estimation of a non-zero source signal, $y_{\text{ID}}(t)$, therefore there is not the possibility of having zero valued messages at those nodes. However, for all the nodes with ID $\in \Theta_Z$, half of the transmitted messages are zero since $y_{\text{ID}}(t) = 0, \forall \text{ID} \in \Theta_Z$. The other half of the the transmitted messages from the nodes with ID $\in \Theta_Z$ is comprised of $W_{k\text{ID}}(t)b_{\text{ID}}(t), k = 1, ..., N$, which are not necessarily zero.

To guarantee having zero valued messages at nodes with ID $\in \Theta_Z$, we consider the following condition:

Condition 3: All the messages that need to be transmitted from the node with ID $\in \Theta_Z$ will be zero valued if $W_{k\text{ID}}(t) = 0, k = 1, ..., N$. In another word the IDth column of the de-mixing matrix, the column that is kept at node ID, should be zero.

A natural solution for satisfying the equality, $W_{k\text{ID}}(t) = 0$, is to encourage sparsity on the elements of de-mixing matrix. Since the M observations of the nodes with ID $\in \Theta_S$ are adequate for separation of all M original source

signals, by encouraging sparsity we expect elimination of the need for observations of the nodes with ID $\in \Theta_Z$ by learning $N - M$ zero valued columns as $W_{k\text{ID}}(t) = 0, k = 1, ..., N$, ID $\in \Theta_Z$.
In order to encourage sparsity on the non-zero elements of the de-mixing matrix, e.g. the weights W_{ij} and W_{ji} on the edge $(i, j) \in \varepsilon$, we can add an l_1 norm regularization term to (6.31) as below:

$$W_{ij} = \arg\min_{W_{ij}} \{I(\mathbf{y}_i, \mathbf{y}_j) + \gamma |W_{ij}|\}, \qquad \forall j \in N(i) \backslash i \tag{6.35}$$

where γ is the regularization parameter.
Making use of the stochastic natural gradient learning rule, evaluation of W_{ij} via (6.35) suggests the following update:

$$W_{ij}(t+1) = W_{ij}(t) - \mu(t)\{\varphi_i[y_i(t)]y_j(t)W_{jj}(t) + \gamma \text{sign}[W_{ij}(t)]\}, \quad i \neq j. \tag{6.36}$$

By using (6.36) and assigning thresholds τ_y and τ_W to the magnitudes of the sources and the elements of the de-mixing matrix respectively, we can refuse of transmitting a message from node i to node j if $|W_{ji}(t)| < \tau_W$ and $|y_i(t)| < \tau_y$. Any unreceived message from node i to node j can be retrieved as zero at node j.
Algorithm 12 shows the distributed BSS algorithm based on the l_1-regularized minimum pairwise mutual information in form of scheme 1 on an arbitrary graph.

6.3.2 Scheme 2

In this section, similar to the Section 6.3.1, we develop distributed algorithms that can be applied for the graphical structure of the network. The distributed algorithms are aimed to be the distributed versions of the centralized mixing and de-mixing matrix based BSS Algorithms 8 and 9. In our proposed distributed algorithms, in the form of scheme 2, each node is aimed to compute the estimates of *all* source signals. To develop distributed algorithms in the form of scheme 2 we follow the following procedures: Firstly, as in common situations there is no prior information about the number of sources, we allocate a source vector of high dimensionality P, to each node of the network, and the algorithm is assumed to separate at most P sources if there are at least P sensor nodes. Secondly, we

Algorithm 12 An on-line de-centralized linear BSS Algorithm based on minimization of l_1-regularized pairwise mutual information (Scheme1-MPMI-l_1) for a time sequence of T time samples.

```
Initialize $W_{ii}(1) = 1, W_{ij}(1) = W_{ji}(1) = 0$ for all $i \in \nu, (i,j) \in \varepsilon$
for $t = 1, ..., T$ do
    if $|W_{mi}(t)| < \tau_W$ & $|y_i(t)| < \tau_y$, $\forall i \in \nu, m \in N(i)\backslash i$ then
        $W_{mi}(t) = 0$ and $y_i(t) = 0$
        node $i$ refuses sending $W_{mi}(t)b_i(t)$ and $y_i(t)$ to node $m$.
    end if
    for all $i \in \nu$ do
        $y_i(t) = \sum_{k \in N(i)} W_{ik}(t)b_k(t)$.
        $W_{mi}(t+1) = W_{mi}(t) - \mu(t)\{\varphi_m[y_m(t)]y_i(t)W_{ii}(t) + \gamma \text{sign}[W_{mi}(t)]\}$,
                                                                        $m \neq i$.
        $W_{ii}(t+1) = W_{ii}(t)$.
    end for
end for
```

split the data across the nodes according to the splitting structure illustrated in Fig. 6.2. Considering the latter procedure we propose the following distributed solutions for on-line linear BSS.

Distributed source estimation

The centralized BSS Algorithms 8 and 9 consist of two steps per observation. In the first step, the sources are estimated given the current observations and the current estimate of the parameters. To estimate the sources in a distributed processing manner, we propose two distributed algorithms where one benefits from a fusion center while the other is fully de-centralized.

Fusion-center based:

The proposed fusion-center based distributed algorithm for source estimation can be applied to both Maximum-Likelihood (ML) and Minimum Average Mutual Information (MAMI) based BSS Algorithms 8 and 9.

Fusion-center approach to ML based BSS:

In the maximum-likelihood based BSS algorithm 8 the sources are estimated as $y(t) = A^{\dagger}(t)b(t)$ which can be represented in the form of a least square problem as $y(t) = \arg\min_y ||A(t)y - b(t)||_2$. Due to the advantageous mathematical properties of quadratic objective functions, such as being differentiable, we consider

$f(y) = \frac{1}{2}||A(t)y - b(t)||_2^2$ as the objective function. Using the splitting structure introduced in Fig. 6.2 we split the objective function as $f(y) = \sum_{i=1}^{N} f_i(y^{\{i\}})$ where $f_i(y^{\{i\}}) = \frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2$, $A_i(t)$ and $b_i(t)$ are the i^{th} row of the matrix $A(t)$ and vector $b(t)$ respectively and $y^{\{i\}}$ is the i^{th} copy of the variable $y_{P\times 1}$ that lives in node i. The optimization problem for distributed source estimation can then be formulated as:

$$\begin{aligned} &\min_{y} \sum_{i=1}^{N} \frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2 \\ &\text{s.t.} \;\; y^{\{i\}} = z \;\text{ for }\; i = 1, ..., N, \end{aligned} \tag{6.37}$$

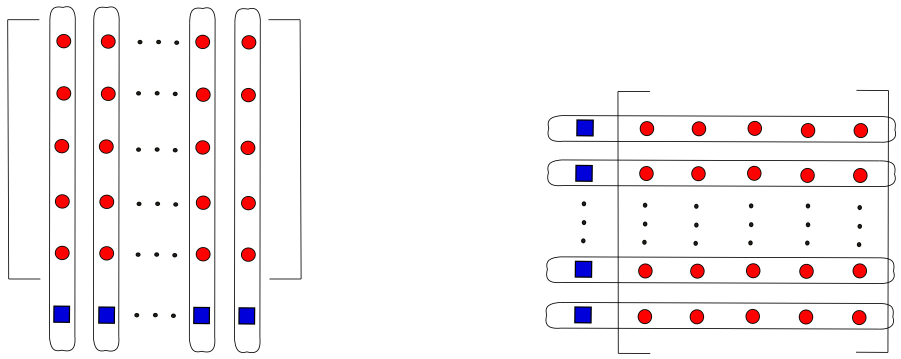
where z is an auxiliary variable.
The augmented Lagrangian [76] for the problem (6.37) can be formulated as:

$$L_\rho(y, z, u) = \sum_{i=1}^{N} \{\frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2 + \frac{\rho}{2}||y^{\{i\}} - z + u_i||_2^2\} \tag{6.38}$$

Making use of the augmented Lagrangian above, a distributed processing solution for the problem (6.37) can be obtained by following the ADMM algorithm [24, 48, 73]:

$$\begin{aligned} y^{\{i,k+1\}}(t) &= \arg\min_{y^{\{i\}}}(\frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2 \\ &+ \frac{\rho}{2}||y^{\{i\}} - z^k + u_i^k||_2^2) \\ z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N}(y^{\{i,k+1\}}(t) + u_i^k) \\ u_i^{k+1} &= u_i^k + y^{\{i,k+1\}}(t) - z^{k+1}. \end{aligned} \tag{6.39}$$

where k denotes the iteration counter.
Fusion-center approach to MAMI based BSS:
According to Algorithm 9, the sources are estimated at each time sample as $y(t) = W(t)b(t)$ which can be represented as a quadratic optimization as $y(t) = \arg\min_y \frac{1}{2}||y - W(t)b(t)||_2^2$. Using the splitting structure introduced in Fig. 6.2, we split the objective function $g(y) = \frac{1}{2}||y - W(t)b(t)||_2^2$ on a column-by-column basis as $g(y) = \sum_{i=1}^{N} g_i(y^{\{i\}})$ where $g_i(y^{\{i\}}) = \frac{1}{2}||y^{\{i\}} - NW^i(t)b_i(t)||_2^2$ and $W^i(t)$ and $b_i(t)$ are the i^{th} column of matrix $W(t)$ and i^{th} row of vector $b(t)$ respectively.



(a) De-mixing matrix (b) Mixing matrix

Figure 6.2: Illustration of the splitting structure of the data. Red circles represent the elements of the mixing and de-mixing matrices. Blue squares represent the observations at each node.

The optimization problem can then be formulated as:

$$\begin{aligned}&\min_{y} \sum_{i=1}^{N} \frac{1}{2}||y^{\{i\}} - NW^{i}(t)b_i(t)||_2^2 \\ &\text{s.t.} \quad y^{\{i\}} = z \text{ for } i = 1, ..., N,\end{aligned} \tag{6.40}$$

where z the auxiliary variable.
The augmented Lagrangian function for the problem (6.40) is formulated as:

$$L_{\rho}(y, z, u) = \sum_{i=1}^{N} \{\frac{1}{2}||y^{\{i\}} - NW^{i}(t)b_i(t)||_2^2 + \frac{\rho}{2}||y^{\{i\}} - z + u_i||_2^2\} \tag{6.41}$$

By implementing the ADMM algorithm on the augmented Lagrangian above, a distributed processing solution for source estimation is obtained:

$$\begin{aligned}y^{\{i,k+1\}}(t) &= \arg\min_{y^{\{i\}}}(\frac{1}{2}||y^{\{i\}} - NW^{i}(t)b_i(t)||_2^2 \\ &+ \frac{\rho}{2}||y^{\{i\}} - z^k + u_i^k||_2^2) \\ z^{k+1} &= \frac{1}{N}\sum_{i=1}^{N}(y^{\{i,k+1\}}(t) + u_i^k) \\ u_i^{k+1} &= u_i^k + y^{\{i,k+1\}}(t) - z^{k+1}.\end{aligned} \tag{6.42}$$

The existence of the variable N in the algorithm above requires prior knowledge about the total number of sensor nodes. In real-world applications where one does not have exact information about the total number of nodes, an approximate value can be used. In those situations a scaled version of the sources will be learned.

By looking to the ADMM updates (6.39) and (6.42) we see that at each iteration there is the need of aggregation of $y^{\{i\}}$ and u_i into a fusion node, calculating the average and then broadcasting the result to each sensor node. Fig. 6.3-a illustrates a sample connectivity pattern of the sensor nodes in fusion-center approach.

De-centralized based:

Its good to note that the fusion-center approach carries similar limitations to the centralized approach, such as high power consumption of the transmission and lack of scalability especially in large-scale problems, since there is the need for communication to a particular location. This motivates us to implement a fully de-centralized algorithm as a distributed solution for source estimation. The de-centralized algorithm that we consider in this paper can be applied to both maximum-likelihood and minimum average mutual information based BSS Algorithms 8 and 9. We first discuss the de-centralized solution for source estimation in ML based BSS Algorithm 8 and then the de-centralized processing of source estimation in MAMI based BSS Algorithm 9.

De-centralized approach to ML based BSS:

Similar to the fusion-center approach we consider $f(y) = \frac{1}{2}||A(t)y - b(t)||_2^2$ as the objective function with the same separability structure as $f(y) = \sum_{i=1}^{N} f_i(y^{\{i\}})$ where $f_i(y^{\{i\}}) = \frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2$. We then consider the following optimization problem over a graph $G = (\nu, \varepsilon)$ with $|\nu| = N$ as:

$$\begin{aligned} \min_{y} \sum_{i\in\nu} \frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2 \\ \text{s.t.} \quad y^{\{i\}} = y^{\{j\}} \ \forall (i,j) \in \varepsilon \end{aligned} \tag{6.43}$$

A de-centralized distributed solution for the problem above can be achieved using distributed algorithms [90, 188, 202] and AMM algorithm which was proposed in Chapter 5. The de-centralization is the consequence of eliminating the global auxiliary variable z. To address (6.43) we use the AMM algorithm because of its simplicity, superior convergence property and its two other major properties; 1) Its asynchronous version does not require any clock for coordination of the nodes. 2) Its synchronous version is optimized in terms of the number of transmitted messages and the required transmission bandwidth. Without loss of generality, in this chapter we only apply the synchronous version of AMM algorithm. As was discussed in Chapter 5, AMM formulates the following regularized augmented Lagrangian function for the problem (6.43) as shown below:

$$\begin{aligned}\hat{L}_{\rho,\alpha}(y,\lambda) =&\sum_{i\in\nu}\{\frac{1}{2}||A_i(t)y^{\{i\}}-b_i(t)||_2^2+\sum_{j\in N(i)\setminus i}[\text{sign}(j-i)\lambda_{j|i}^T(y^{\{j\}}-y^{\{i\}})\\&+\frac{\rho}{2}||y^{\{i\}}-y^{\{j\}}||_2^2-\frac{1}{(2\alpha)}||\lambda_{i|j}-\lambda_{j|i}||_2^2]\},\end{aligned} \tag{6.44}$$

where $\lambda_{i|j}$ is the dual variable that is held at node i and is related to the constraint on the edge (i,j). The solution to the problem (6.43) can be obtained by finding the saddle point of the augmented Lagrangian 6.44 as:

$$(y^*,\lambda^*)=\arg\max_{\lambda}\min_{y}\hat{L}_{\rho,\alpha}(y,\lambda) \tag{6.45}$$

where (y^*,λ^*) is the saddle point.

Sync-AMM algorithm leads to the following updates for approaching to the saddle point:

$$\begin{aligned}y^{\{i,k+1\}}(t) =&\arg\min_{y^{\{i\}}}\{\frac{1}{2}||A_i(t)y^{\{i\}}-b_i(t)||_2^2+\beta_i^{k^T}y^{\{i\}}+\sum_{j\in N(i)\setminus i}\frac{\rho}{2}||y^{\{i\}}-y^{\{j,k\}}(t)||_2^2\}\\ \beta_i^{k+1} =&\beta_i^{k-1}+\alpha\sum_{j\in N(i)\setminus i}(2y^{\{i,k\}}(t)-y^{\{j,k-1\}}(t)-y^{\{j,k+1\}}(t)),\end{aligned} \tag{6.46}$$

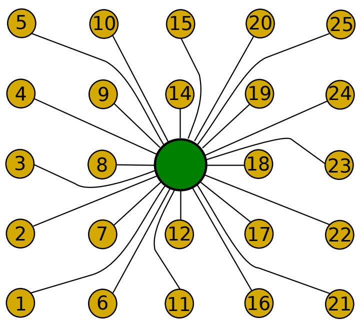
where $\beta_i^k=-\sum_{j\in N(i)\setminus i}\text{sign}(j-i)\lambda_{j|i}^k$.

De-centralized approach to MAMI based BSS:

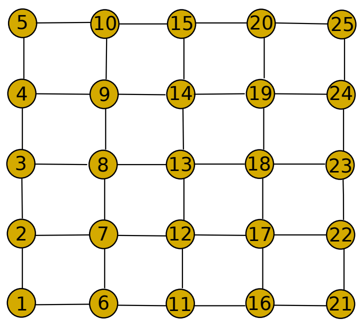
By using $g(y)=\frac{1}{2}||y-W(t)b(t)||_2^2$ as the objective function and considering the same splitting structure mentioned earlier as $g(y)=\sum_{i=1}^N g_i(y^{\{i\}})$ where $g_i(y^{\{i\}})=\frac{1}{2}||y^{\{i\}}-NW^i(t)b_i(t)||_2^2$ and formulating an optimization problem over a graph $G=(\nu,\varepsilon)$ with $|\nu|=N$ as:

$$\begin{aligned}&\min_{y}\sum_{i\in\nu}\frac{1}{2}||y^{\{i\}}-NW^i(t)b_i(t)||_2^2\\&\text{s.t. } y^{\{i\}}=y^{\{j\}}\ \forall(i,j)\in\varepsilon\end{aligned} \tag{6.47}$$

a de-centralized processing solution for source estimation can be obtained by



(a) Fusion-center



(b) De-centralized

Figure 6.3: Illustration of a sample connectivity pattern of the graph of scheme 2.

formulating the regularized augmented Lagrangian function as:

$$\hat{L}_{\rho,\alpha}(y,\lambda) = \sum_{i\in\nu}\{\frac{1}{2}||y^{\{i\}} - NW^i(t)b_i(t)||_2^2 + \sum_{j\in N(i)\backslash i}[\text{sign}(j-i)\lambda_{j|i}^T(y^{\{j\}} - y^{\{i\}}) + \frac{\rho}{2}||y^{\{i\}} - y^{\{j\}}||_2^2 - \frac{1}{(2\alpha)}||\lambda_{i|j} - \lambda_{j|i}||_2^2]\}, \tag{6.48}$$

and finding its saddle point via Sync-AMM algorithm as:

$$\begin{aligned} y^{\{i,k+1\}}(t) =& \arg\min_{y^{\{i\}}}\{\frac{1}{2}||y^{\{i\}} - NW^i(t)b_i(t)||_2^2 + \beta_i^{k^T} y^{\{i\}} + \sum_{j\in N(i)\backslash i}\frac{\rho}{2}||y^{\{i\}} - y^{\{j,k\}}(t)||_2^2\} \\ \beta_i^{k+1} =& \beta_i^{k-1} + \alpha\sum_{j\in N(i)\backslash i}(2y^{\{i,k\}}(t) - y^{\{j,k-1\}}(t) - y^{\{j,k+1\}}(t)), \end{aligned} \tag{6.49}$$

Fig. 6.3-b illustrates a sample connectivity pattern of the sensor nodes in de-centralized approach.

Heuristic based on distributed averaging:

An alternative method for de-centralized processing of source estimation in MAMI based BSS Algorithm 9 is to convert the constraint optimization problem (6.47) into a distributed averaging problem. This can be achieved by considering a network of N nodes where each node has an initial value as $y^{\{i,1\}}(t) = NW^i(t)b_i(t)$ and the objective is to estimate the average value $y(t) = \frac{1}{N}\sum_{i=1}^N NW^i(t)b_i(t)$ at each node by only communicating with the neighbouring nodes.

There exist many distributed averaging methods such as [59,126,133,191]. Among the latter methods, [191] reveals superior convergence. Making use of [191] in

Algorithm 13 An on-line de-centralized linear BSS Algorithm based on maximization of likelihood (Scheme2-ML) for a time sequence of T time samples. At each time t, the algorithm estimates the source samples as the consensus of $y^{\{i\}}(t)$ given $A(t)$ and observations $b(t)$, and then updates the mixing matrix $A(t)$.

```
Initialize $A_i, i \in \nu$
for $t = 1, ..., T$ do
    Initialize $y^{\{i,1\}}(t) = \beta_i^0 = \beta_i^1 = 0 \ \forall i \in \nu, \ k \leftarrow 1$
    do [in parallel]
        $y^{\{i,k+1\}}(t) = \arg\min_{y^{\{i\}}} \{\frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2 + \beta_i^{k^T} y^{\{i\}} +$
                $\sum_{j \in N(i)\backslash i} \frac{\varrho}{2}||y^{\{i\}} - y^{\{j,k\}}(t)||_2^2\}$
        $\beta_i^{k+1} = \beta_i^{k-1} + \alpha \sum_{j \in N(i)\backslash i}(2y^{\{i,k\}}(t) - y^{\{j,k-1\}}(t) - y^{\{j,k+1\}}(t))$
        $k \leftarrow k+1$
    while $h^k >= \epsilon_p$
    for all $i \in \nu$ [in parallel] do
        $A_i(t+1) = A_i(t) - \mu(t)A_i(t)F[y^{\{i,k-1\}}(t)]$
    end for
end for
```

our problem, suggests the following updates for estimating $y(t)$ at each node over a graph $G = (\nu, \varepsilon)$ as:

$$\begin{aligned} y_p^{\{i,k+1\}}(t) &= H_{ii} y_p^{\{i,k\}}(t) + \sum_{j \in N(i)\backslash i} H_{ij} y_p^{\{j,k\}}(t) \\ \text{where} \quad y^{\{i\}}(t) &= [y_1^{\{i\}}(t), ..., y_P^{\{i\}}(t)]^T \end{aligned} \tag{6.50}$$

where H is output of the following optimization:

$$\begin{aligned} &\min_H ||H - (1/N)\mathbf{1}\mathbf{1}^T||_2 \\ &\text{s.t.} \quad H \in \mathcal{S} \ , \ H = H^T \ , \ H\mathbf{1} = \mathbf{1} \end{aligned} \tag{6.51}$$

where $\mathcal{S} = \{H \in \mathbb{R}^{N \times N} | H_{ij} = 0 \text{ if } (i,j) \notin \varepsilon \text{ and } i \neq j\}$ and $\mathbf{1}$ denotes a vector with all coefficients one.

Using (6.50) instead of (6.49) requires lower communication bandwidth and simplifies the computational complexity. However, the Lagrangian-based algorithm (6.49) can sustain communication link failure and displays faster convergence in graphs with high degree of sparsity in comparison with distributed averaging (6.50). The distributed averaging (6.50) cannot sustain communication link fail-

Algorithm 14 An on-line de-centralized linear BSS Algorithm based on minimization of average mutual information (Scheme2-MAMI) for a time sequence of T time samples. At each time t, the algorithm estimates the source samples as the consensus of $y^{\{i\}}(t)$ given $W(t)$ and observations $b(t)$, and then updates the de-mixing matrix $W(t)$.

```
Initialize $W_i, i \in \nu$
for $t = 1, ..., T$ do
    Initialize $y^{\{i,1\}}(t) = \beta_i^0 = \beta_i^1 = 0 \ \forall i \in \nu, \ \ k \leftarrow 1$
    do [in parallel]
        $y^{\{i,k+1\}}(t) = \arg\min_{y^{\{i\}}} \{\frac{1}{2}||y^{\{i\}} - NW^i(t)b_i(t)||_2^2 + \beta_i^{k^T} y^{\{i\}} +$
                $\sum_{j \in N(i) \setminus i} \frac{\varrho}{2}||y^{\{i\}} - y^{\{j,k\}}(t)||_2^2\}$
        $\beta_i^{k+1} = \beta_i^{k-1} + \alpha \sum_{j \in N(i) \setminus i}(2y^{\{i,k\}}(t) - y^{\{j,k-1\}}(t) - y^{\{j,k+1\}}(t))$
        $k \leftarrow k + 1$
    while $h^k >= \epsilon_p$
    for all $i \in \nu$ [in parallel] do
        $W^i(t+1) = W^i(t) + \mu(t)\acute{F}[y^{\{i,k-1\}}(t)]W^i(t)$
    end for
end for
```

ure as the deduced matrix H would no longer satisfy the constraint $H\mathbf{1} = \mathbf{1}$. This is in contrast to the Lagrangian-based algorithm (6.49) that requires the deduced subgraph to be only connected. So depending on the availability of resources in a given network one can use either (6.50) or (6.49) as a de-centralized solution for source estimation in MAMI based BSS Algorithm 9.

Distributed parameter update

ML based BSS:

After estimating the sources, by preserving the splitting structure described in Fig. 6.2 each node can update its own parameter without the need for any communication with the neighbour nodes as:

$$A_i(t+1) = A_i(t) - \mu(t)A_i(t)(\Lambda^{\{i\}}(t) - \psi[y^{*\{i\}}(t)]y^{*\{i\}}(t)^T) \tag{6.52}$$

where $\Lambda^{\{i\}}(t) = \text{diag}[\text{diag}[\psi[y^{*\{i\}}(t)]y^{*\{i\}}(t)^T]]$.

MMI based BSS:

By preserving the splitting structure shown in Fig. 6.2 each node can update its own parameter locally as follows:

$$W^i(t+1) = W^i(t) + \mu(t)(\acute{\Lambda}^{\{i\}}(t) - \varphi[y^{*\{i\}}(t)]y^{*\{i\}}(t)^T)W^i(t) \tag{6.53}$$

where $\acute{\Lambda}^{\{i\}}(t) = \text{diag}[\text{diag}[\varphi[y^{*\{i\}}(t)]y^{*\{i\}}(t)^T]]$.

Algorithms 13 and 14 show the distributed BSS algorithm based on ML and MAMI in the form of scheme 2 respectively where $h^k = \sum_{i\in\nu}\sum_{j\in N(i)\setminus i}\frac{1}{2}||y^{\{i,k\}}(t) - y^{\{j,k\}}(t)||_2$, $\epsilon_p = \sqrt{P}\epsilon^{abs} + \epsilon^{rel}\max_{i\in\nu}\{||y^{\{i,k\}}(t)||_2\}$ and $\epsilon^{abs} > 0$ and $\epsilon^{rel} > 0$.

Convergence Analysis:

Finding a solution for the constrained problems (6.37) and (6.43) requires following the algorithms (6.39) and (6.46) in the form of a min-max optimization. However, the minimization steps in (6.39) and (6.46) have a solution set in a $P-1$ dimensional space. This is because the function $f_i(y^{\{i\}}) = \frac{1}{2}||A_i(t)y^{\{i\}} - b_i(t)||_2^2$ is a convex function with a $P-1$ dimensional solution set. Therefore the minimization steps in (6.39) and (6.46) may not be highly informative about the direction of optimization towards the saddle point of the augmented Lagrangian function in the early iterations. In fact the ambiguity in finding the exact solution among all possible solution set vanishes when the constraints in (6.37) and (6.43) are satisfied and that is attained when the iteration evolves. Thus the convexity of the objective function results in a sub-linear convergence rate for source estimation in the mixing matrix based BSS algorithms. The rate of convergence is expected to decrease by increasing the dimensionality of the source vector, P. The sub-linear convergence rate of the ADMM based algorithms for general convex functions has been determined in [115].

In contrast, finding the solution for the problems (6.40) and (6.47) requires following a min-max algorithm in which the minimization is performed over a *strictly* convex function in the form of $g_i(y^{\{i\}}) = \frac{1}{2}||y^{\{i\}} - NW^i(t)b_i(t)||_2^2$ with a unique minimizer in each node. Therefore the minimization steps in (6.42) and (6.49) give the highest possible information about the direction towards the saddle point of the augmented Lagrangian function. This results in a linear convergence rate in source estimation in de-mixing matrix based BSS algorithms where the convergence rate is irrespective to the dimensionality of the source vector. The linear convergence rate of the ADMM based algorithms for strictly convex functions has been determined in [43].

To compare the convergence rate of algorithm (6.49) with the distributed averaging algorithm (6.50) in terms of the degree of sparsity of the graph, we recall the per-iteration convergence factor, r_{iter}, and its associated convergence time, τ_{iter}, for distributed averaging that are defined in [191] as:

$$r_{\text{iter}} = \sup_{x_p^{\{k\}} \neq \bar{x}_p} \frac{||x_p^{\{k+1\}} - \bar{x}_p||_2}{||x_p^{\{k\}} - \bar{x}_p||_2} = ||H - (1/N)\mathbf{1}\mathbf{1}^T||_2, \tag{6.54}$$

where $x_p^{\{k\}} = [y_p^{\{1,k\}}, ..., y_p^{\{N,k\}}]^T$ for $p = 1, ..., P$ in our BSS problem and

$$\tau_{\text{iter}} = \frac{1}{\log(1/r_{\text{iter}})}. \tag{6.55}$$

From (6.54) and (6.55) we see that the convergence time has a direct relationship with the value of $||H - (1/N)\mathbf{1}\mathbf{1}^T||_2$. Increasing the sparsity of a graph is equivalent to decreasing the degree of freedom, the number of non-zero elements of H, in correct estimation of H as a minimizer of $||H - (1/N)\mathbf{1}\mathbf{1}^T||_2$ and therefore increasing the convergence time. This is in contrast to the algorithm (6.49) as increasing the sparsity of the graph is equivalent to decreasing the number of constraints and their corresponding dual variables which consequently decreases the dimensionality of the search space for finding the saddle point of the augmented Lagrangian function.

6.3.3 Required output power

One of the major factors that influences the WSN design is the power consumption. Power consumption can be divided to three domains: sensing, data processing and communication. According to [4], the maximum power is consumed in data communication. [186] approximated the required output power for a reliable data transmission between the nodes of the WSN as:

$$P_{Tx}(d) = \epsilon q^{\eta} \tag{6.56}$$

where ϵ is a constant that depends on the characteristic of the transmitting and receiving antennas, q denotes the range of transmission and η is the path loss exponent which is about 2 for free space. Equation (6.56) allows the evaluation of the power consumption of the sensor nodes when the proposed distributed BSS algorithms are applied.

Table 6.1 compares the properties of the distributed *de-centralized* BSS algo-

Table 6.1: Properties of the proposed distributed de-centralized BSS algorithms.

	S1-MAMI	S1-MPMI	S1-MPMI-l_1	S2-ML	S2-MAMI
CB	256 bit/t	256 bit/t	256 bit/t	$128P$ bit/k	$128P$ bit/k
CC	$O(\|N(i)\|)$	$O(\|N(i)\|)$	$O(\|N(i)\|)$	$O(P^3 + P\|N(i)\|)$	$O(5\|N(i)\| + 4P)$
CS	1 per t	1 per t	1 per t	k per t	k per t
EP	Yes	No	No	Yes	Yes
PC	$O(N\epsilon\bar{q}^{\eta})$	$O(N\epsilon\bar{q}^{\eta})$	$O(M\epsilon\bar{q}^{\eta})$	$O(kN\epsilon\bar{q}^{\eta})$	$O(kN\epsilon\bar{q}^{\eta})$
PG	Non-uniform	Non-uniform	Non-uniform	Uniform	Uniform
S/A	Sync in t	Sync in t	Sync in t	(Sync in t) (Sync/Async in k)	(Sync in t) (Sync/Async in k)

rithms that was proposed in this chapter. We used the following notations in the table, CB: The maximum required communication bandwidth per communication channel, where we assumed that 64 bit is required to represent a scalar. CC: The computational complexity at the i^{th} node of the network. CS: The required communication steps. EP: Equivariant property. PC: The total power consumption of the network at each time sample t. PG: The performance of the algorithm on graphs with different connectivity pattern. S/A: Synchronous or Asynchronous.

6.4 Experimental Results

To evaluate the performance of the proposed distributed BSS algorithms we simulated a low noise delay-restricted environment where the observations are obtained from a linear instantaneous mixture of the sources. To set up a connectivity pattern over the sensor nodes we used the d_m-neighbourhood graph strategy in which each node is connected to other nodes if their pairwise distances are smaller than or equal to d_m.

6.4.1 Experimental Setup

In the experiments four source signals along with 25 sensor nodes were used. Fig. 6.4 shows the position of the sensor nodes and the sources. A total of 6 different connectivity patterns (via setting d_m, where d_m denotes the threshold of friendship between the nodes, to different values) were also used in the

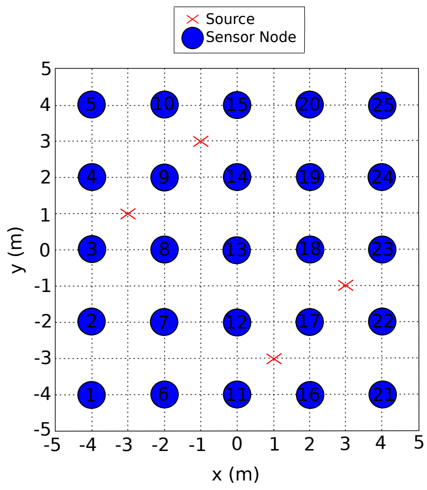


Figure 6.4: Illustration of the position of sensor nodes and sources.

experiments.
Speech signals of 6 seconds in length with the variances equal to one were used as the original source signals. The speech signals were sampled with a sampling frequency of 16 kHz. The observations of the sensor nodes were corrupted by zero mean Gaussian noises with variances equal to 0.02. The elements of the original mixing matrix were created by sampling from a zero mean unit variance normal distribution. The parameters of the hyperbolic tangent ϑ and θ were set to 1 and 100 respectively.

Scheme 1

For the case of distributed BSS algorithms in the form of scheme 1, the identity matrix was used as an initial value for the de-mixing matrix $W_{25\times25}$. A constant step size $\mu = 1 \times 10^{-4}$ was used for Scheme1-MAMI, Scheme1-MPMI and Scheme1-MPMI-l_1 algorithms. The regularization parameter γ and the thresholds τ_y and τ_W were set to 0.02, 0.01 and 1×10^{-4} respectively. For the case of Scheme1-MPMI-l_1 algorithm, no noise was considered at the sensor nodes.

Scheme 2

For the case of distributed BSS algorithms in the form of scheme 2, the number of sources where over estimated at each node as 6 and a mixing and a de-mixing matrix of dimension 25×6 and 6×25 were used respectively. The initial values

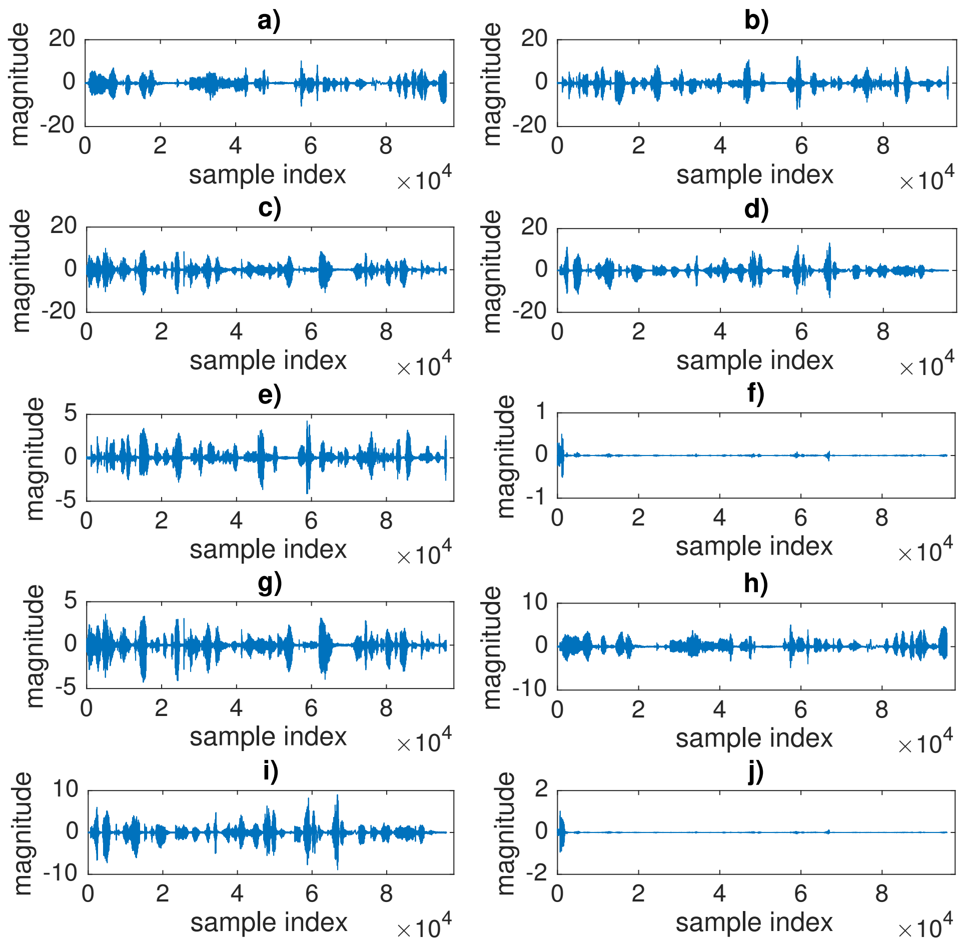


Figure 6.5: Visualization of the original and separated signals. (a-d) Original source signals. (e-j) Separated signals.

of the elements of the mixing and de-mixing matrices were obtained by sampling from a zero mean and unit variance normal distribution. A constant step size μ equal to 2×10^{-3} and 1×10^{-4} was used for ML and MAMI based BSS algorithms respectively. The parameters ρ and α were set to 1 and 0.5 respectively. We used $\epsilon^{abs} = 1\times10^{-10}$ and $\epsilon^{rel} = 1\times10^{-10}$ as a stopping criteria to terminate the iterations for source estimation.

6.4.2 Results

To assess the performance of the maximum-likelihood based BSS algorithm in source separation when the number of sources is overestimated, we applied Al-

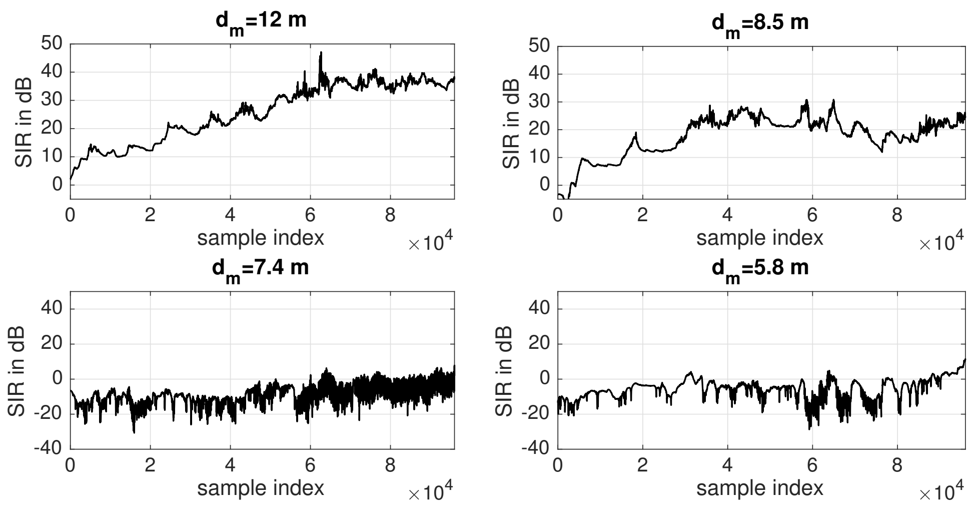
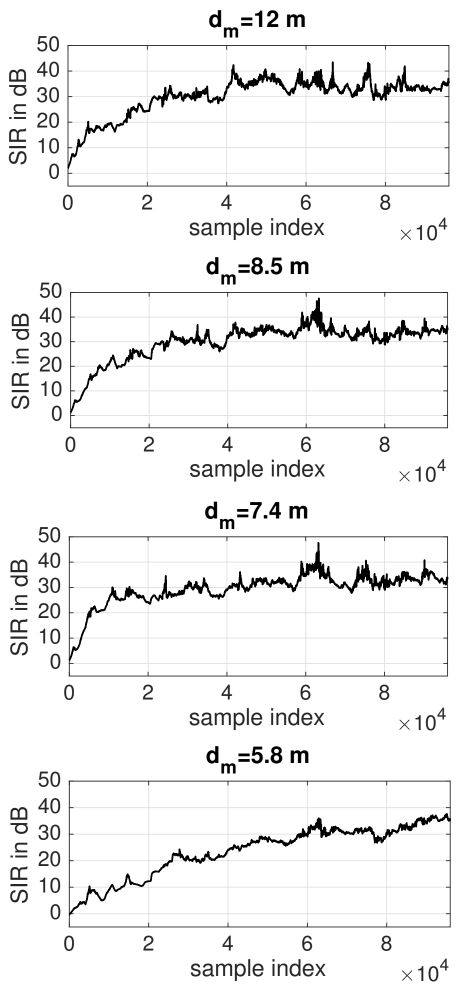


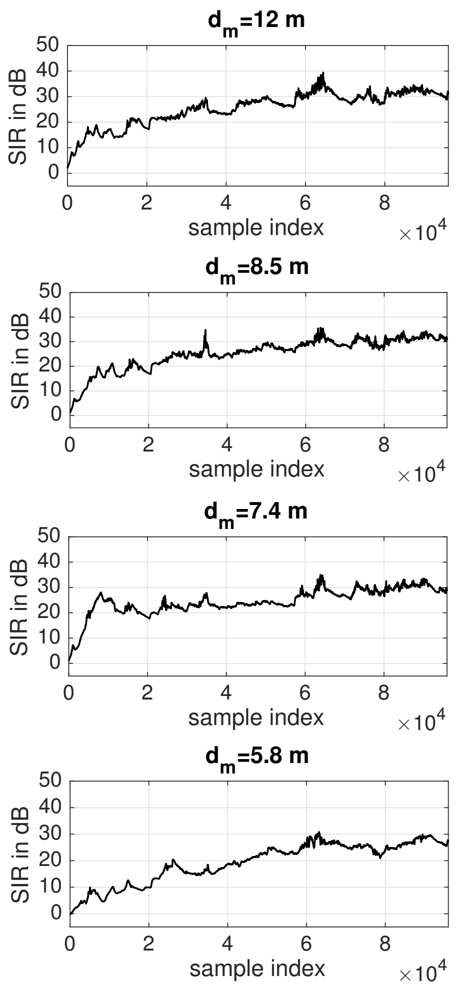
Figure 6.6: Illustration of the average SIR of the non-zero estimated signals using Scheme1-MAMI algorithm.

gorithm 8 in a centralized processing manner and in a noiseless scenario, where we used the setup of scheme 2, and the result of separation is illustrated in Fig. 6.5. From Fig. 6.5 we see that out of six estimated sources two of them are zero output.

Fig. 6.6 shows the average Signal to Interference Ratio (SIR) of the non-zero estimated signals across time when the Scheme1-MAMI algorithm is applied for different d_m. From the results in Fig. 6.6 we see a significant degradation of the performance of Scheme1-MAMI algorithm over graphs with high degree of sparsity. Fig. 6.7 shows the average SIR of the non-zero estimated signals across time when Scheme1-MPMI and Scheme1-MPMI-l_1 algorithms are applied for different d_m. Implementation of Scheme1-MPMI and Scheme1-MPMI-l_1 algorithms over the graphs with $d_m = 12$ m led to estimation of the sources at only the node 1, 14, 17 and 23 while over the graphs with $d_m = 8.5$ m, $d_m = 7.4$ m and $d_m = 5.8$ m led to estimation of the sources at the aforementioned nodes along with some other nodes of the network. From the results in Fig. 6.7 we see the good performance of the algorithms over sparse graphs. Table 6.2 compares the l_1 norm of the columns of the de-mixing after implementation of Scheme1-MPMI and Scheme1-MPMI-l_1 algorithms on a fully connected graph ($d_m = 12$ m) at time $t = 6s$, where we excluded $W_{ii}, i = 1, ..., 25$ from W when calculated the norms. To make a fair comparison, the results in Table 6.2 are obtained by eliminating



(a) Scheme1-MPMI



(b) Scheme1-MPMI-l_1

Figure 6.7: Illustration of the average SIR of the non-zero estimated signals using Scheme1-MPMI and Scheme1-MPMI-l_1 algorithms.

Table 6.2: l_1-norm of the columns of the de-mixing matrix W, where $W_{ii}, i = 1, ..., 25$ are excluded. Here the value that corresponds to W^E is the average norm of all the columns excluding W^1, W^{14}, W^{17} and W^{23}.

Column	W^1	W^{14}	W^{17}	W^{23}	W^E
Scheme1-MPMI	3.0096	3.6087	2.8780	4.1454	1.1036
Scheme1-MPMI-l_1	7.0968	8.2470	6.7427	8.7030	0.0361

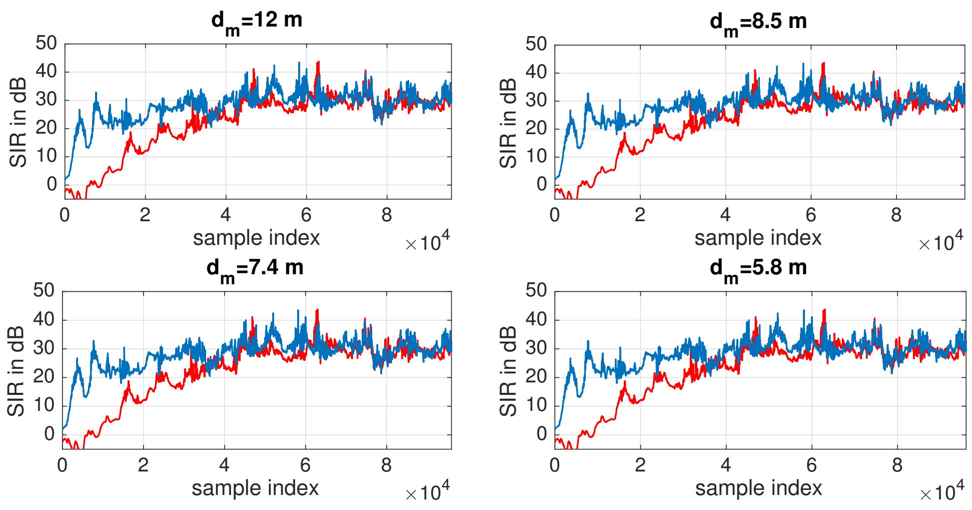


Figure 6.8: Illustration of the average SIR of the non-zero estimated signals using Scheme2-ML and Scheme2-MAMI algorithms. Red) Scheme2-ML. Blue) Scheme2-MAMI.

the noise at the sensor nodes for the case of Scheme1-MPMI algorithm. From the results in Table 6.2 we see that all the elements of the de-mixing matrix, excluding $W_{ii}, i = 1, ..., 25$, that are kept at the nodes which estimate the zero signals take small values when Scheme1-MPMI-l_1 is applied.

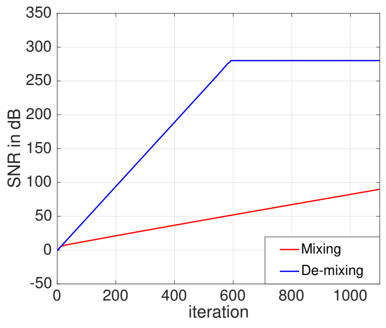
In Fig. 6.8 the average SIR of the non-zero estimated signals is plotted across

Table 6.3: The required RF output power over different graphs.

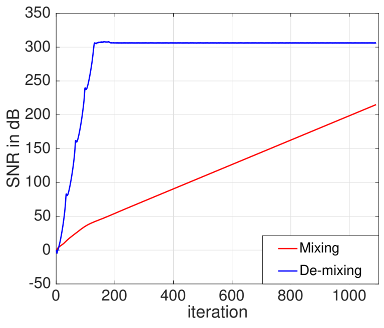
d_m	4.5	5.8	7.4	8.5	12
Scheme1-MAMI	506ϵ	800ϵ	1280ϵ	1664ϵ	2160ϵ
Scheme1-MPMI	506ϵ	800ϵ	1280ϵ	1664ϵ	2160ϵ
Scheme1-MPMI-l_1	411ϵ	578ϵ	861ϵ	994ϵ	1155ϵ

Table 6.4: The required RF output power for different target MSEs over a graph with $d_m = 4.5\ m$.

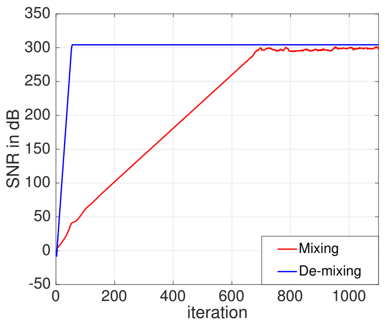
MSE	10^{-5}	10^{-10}	10^{-15}	10^{-20}
Scheme2-MAMI	38962ϵ	66286ϵ	93104ϵ	120428ϵ
Scheme2-ML	66792ϵ	212014ϵ	452870ϵ	694232ϵ



(a) De-centralized $d_m = 12$ m



(b) De-centralized $d_m = 2$ m



(c) Fusion-center

Figure 6.9: Comparative performance of mixing and de-mixing matrix based BSS algorithms in distributed source estimation in the form of scheme 2.

time when Scheme2-ML and Scheme2-MAMI algorithms are applied for different d_m. From the results in Fig. 6.8 we see a uniform performance of the algorithms over different graphs.

To compare the convergence rate of the distributed mixing and de-mixing matrix based BSS algorithms in distributed source estimation, in the form scheme 2, we plotted the average SNR of the estimated sources over the sensor nodes for the two algorithms at time sample 48000 in the utterances ($t = 3s$) as shown in Fig. 6.9, where we used the definition $\text{SNR}_i = 10\log_{10}(\frac{||A^{\dagger}(t)b(t)||_2^2}{||A^{\dagger}(t)b(t)-y^{\{i,k\}}(t)||_2^2})$ and $\text{SNR}_i = 10\log_{10}(\frac{||W(t)b(t)||_2^2}{||W(t)b(t)-y^{\{i,k\}}(t)||_2^2})$ for mixing and de-mixing matrix based BSS algorithms respectively. From the results in Fig. 6.9 we see a quicker rate of convergence when the de-mixing matrix based BSS algorithm is applied in comparison with the mixing matrix based BSS algorithm.

Fig. 6.10 compares the performance of the Lagrangian-based update (6.49) and distributed averaging update (6.50) in distributed source estimation over a graph

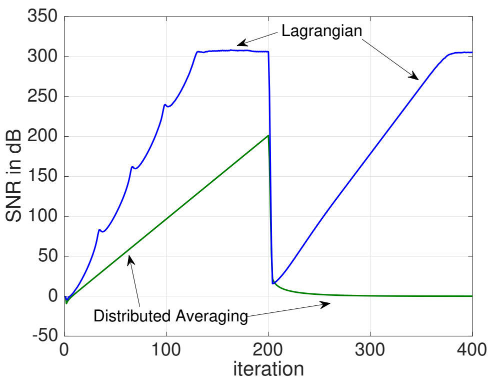


Figure 6.10: Illustration of the average SNR of the estimated signals in communication link failure scenario over a graph with $d_m = 2$.

with $d_m = 2$ when the network encounters with communication link failure. In Fig. 6.10 it is assumed that the edges (4,5) and (13,18) are missed after 200 iterations at the time sample 48000. From the results in Fig. 6.10 we see that the Lagrangian-based update (6.49) can adapt itself to the new graph which is in contrast to the distributed averaging update (6.50).

In Table 6.3, the average required transmission power for source estimation per time sample over different graphs is shown when Scheme1-MAMI, Scheme1-MPMI and Scheme1-MPMI-l_1 algorithms are applied. Table 6.4 shows the required transmission power for source estimation at the time sample 48000 for reaching to the target MSEs using Scheme2-ML and Scheme2-MAMI algorithms, where we used the definition MSE $= \frac{1}{N}\sum_{i=1}^{N} ||A^{\dagger}(t)b(t) - y^{\{i,k\}}(t)||_2^2$ and MSE $= \frac{1}{N}\sum_{i=1}^{N} ||W(t)b(t) - y^{\{i,k\}}(t)||_2^2$ for ML and MAMI based BSS algorithms respectively.

6.5 Summary

In this chapter we introduced two schemes for in-network processing of adaptive linear BSS. The minimum transmission power is attained when the scheme 1 is applied over the de-mixing matrix based BSS algorithms. However the separation performance of the BSS algorithms that are applied in the form of scheme 1 depend on the connectivity pattern of the graph of the network. On the contrary, distributed implementation of the BSS algorithms in the form of scheme 2 allows to achieve a uniform separation performance over any arbitrary connected graph. It has also been observed that a quicker rate of convergence for distributed source estimation is obtained over the de-mixing matrix based BSS algorithms in comparison with the corresponding mixing matrix based BSS algorithms in both schemes. In addition the scheme 1 is more scalable than scheme 2 as there is no need for a consensus across all the nodes in the network when the scheme 1 is applied.

7 Conclusions

This dissertation addressed a novel research topic in engineering, Distributed Blind Source Separation [79, 98]. Blind source separation has a variety of applications such as in Audio Signal Processing [184], Biomedical [122, 183], Astronomy [140] and Finance [16, 119]. This thesis focused on the application of BSS in audio signal processing for source tracking over sensor networks in a distributed processing manner. The BSS algorithms that were considered in this thesis are based on statistical methods that investigate mixing or de-mixing models which maximize the independency of the estimated sources as a solution for separation. Making use of the stochastic gradient method, we were able to develop adaptive algorithms for source tracking in a real-time processing manner. We used the natural gradient [8] extension of the adaptive algorithm as it was more stable.
In the context of blind source separation, the main focus of this thesis was on the linear generative models in which the observations are obtained from the linear instantaneous mixture of the sources. Therefore, direct implementation of the proposed distributed BSS algorithms is suited for scenarios in which there is a linear multiplicative mixing model between the recordings and the sources. An example scenario is source tracking in a cocktail party in an anechoic room.
We investigated two schemes for distributed BSS over sensor networks. In scheme 1, for the case of *de-mixing* matrix based BSS algorithm, each sensor node was

responsible for extracting one source via receiving a weighted sum of the observations of its neighbour nodes. Using scheme 1 facilitates the instantaneous estimation of the sources without requiring iterative procedures for the case of *de-mixing* matrix based BSS algorithm. Eliminating the iterative procedures led to achieving low transmission power for source tracking over wireless sensor networks. By applying the first scheme over a well designed sparse network we were also able to eliminate the need for transmission to long distances. However, the separation performance of the BSS algorithms that are applied in the form of scheme 1 depend on the connectivity pattern of the graph of the network.

In the second scheme, we formulated a consensus problem for source estimation. This allowed an estimation of all sources in each sensor node. In contrast to the scheme 1, the separation performance of the scheme 2 was irrespective to the connectivity pattern of the graph of the network. In fact its performance is analogous to the corresponding centralized algorithm. However, scheme 2 requires an iterative procedure for source estimation, which lead to a higher transmission power compared to the scheme 1. In comparison with the traditional centralized processing approaches, the proposed distributed de-centralized schemes eliminated the need for transmission to a centralized location.

In overall, the *de-mixing* matrix based BSS algorithms, that we considered in this thesis outperform the *mixing* matrix based BSS algorithms when are implemented in a distributed processing manner. Applying the *de-mixing* based BSS algorithm in the form of scheme 2 leads to a *strictly* convex optimization problem for source estimation at a given parameter matrix. The *strict* convexity of the objective function of the problem results in a quicker rate of convergence of distributed ADMM based algorithms in comparison with the optimization problems with convex objective functions. If fewer iterations are required for the convergence of the ADMM based algorithms, then less transmission power will be consumed. Therefore, the most obvious line of future work is to accelerate the convergence of the proposed AMM algorithm. There are a variety of accelerated gradient and ADMM based algorithms that can be used as a starting point, e.g. [19, 60, 62, 95, 138, 141].

The performance of the adaptive BSS algorithms that we considered in this thesis depends on the distribution of the original signals. In order to make the algorithm suitable for the Laplacian distributions we had to choose the appropriate activation function, e.g. $\tanh(.)$. This may limit the performance of the proposed distributed BSS algorithm to a particular set of source signals. Thus, a future line of the work is to apply the proposed distributed schemes on the adaptive BSS algorithms that can be used for any distribution of the sources. Example references for the universally convergent adaptive BSS algorithms can be found in [9, 110, 199].

Some of the work presented in this dissertation concentrated on improvements of distributed algorithms for linearly constrained convex optimization problems. We considered two forms of problems. For the problems in which all primal variables are coupled by a global single block linear constraint, the improvements include: eliminating the need for a master node (centralized location), reduced communication requirements and an asynchronous updating scheme. For the problems with a separable constraint, the improvements include: reduced computational cost, reduced transmission bandwidth and the elimination of the need for any coordination between the nodes of a network. Since many problems can be cast into linearly constrained convex optimization problems, the improvements that were presented in this thesis can be applied in a variety of applications such as in signal an image processing, machine learning and statistics.

The distributed BSS algorithms that were presented in this thesis can easily be applied to the scenarios in which the observations are obtained from a convolutive mixture of the sources. To achieve the latter, the observed data need to be first transferred to the frequency domain via Fourier Transform to produce a multiplicative model for each frequency bin. By making use of the aforementioned Time-Frequency domain approach, we are able to apply the presented methods for distributed source separation in cocktail parties in *reverberant* environments. However, using the Time-Frequency domain approach may result in permutation and scaling problems which can be addressed by the proposed methods in [53, 88, 165, 178]. A future line of work is to use the permutation and scaling solutions along with our proposed distributed methods to develop distributed algorithms for source tracking in convolutive models.

Appendices

A

A.1 Proof of the inequality in (5.30) and (5.31)

Since $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$ are pointwise minimum of an affine function of $\lambda_{j|i}$ and $\lambda_{i|j}$ respectively, they are concave [25, chapter 5]. Accordingly, we denote by C_i and C_j the optimal set, the set of all maximizers, of $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$ respectively. We then denote by $\lambda^o_{j|i} \in C_i$ and $\lambda^o_{i|j} \in C_j$ one of the optimal points of the functions $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$ that are evaluated as:

$$\begin{aligned}\lambda^o_{j|i} &= \arg\max_{\lambda_{j|i}} g_i(\lambda_{j|i}) \\ \lambda^o_{i|j} &= \arg\max_{\lambda_{i|j}} g_j(\lambda_{i|j}).\end{aligned} \tag{A.1}$$

According to the basic property of the maximum [161,197], we have the following relationship for any arbitrary functions $g_i(.)$ and $g_j(.)$

$$\max_{\lambda} \{g_i(\lambda) + g_j(\lambda)\} \leq \max_{\lambda_1} g_i(\lambda_1) + \max_{\lambda_2} g_j(\lambda_2). \tag{A.2}$$

The equality in (A.2) holds when $C_i \cap C_j \neq \varnothing$, where $\varnothing$ denotes the empty set. According to (5.29) and by making use of (A.1) and (A.2) we can establish the following relationship:

$$\begin{aligned}\max_{\lambda}\{g_i(\lambda)+g_j(\lambda)\} &= \max_{\lambda} g_{ij}(\lambda)\\ &= g_{ij}(\lambda^*_{ij}) \leq g_i(\lambda^o_{j|i})+g_j(\lambda^o_{i|j}).\end{aligned} \tag{A.3}$$

We now recall the global dual function $\hat{g}_{ij}(\lambda_{i|j},\lambda_{j|i})$ from which $\lambda^*_{i|j}$ and $\lambda^*_{j|i}$ are evaluated. In fact, $\lambda^*_{i|j}$ and $\lambda^*_{j|i}$ are obtained as:

$$\begin{aligned}(\lambda^*_{i|j},\lambda^*_{j|i}) &= \arg\max_{\lambda_{i|j},\lambda_{j|i}} \hat{g}_{ij}(\lambda_{i|j},\lambda_{j|i})\\ &= \arg\max_{\lambda_{i|j},\lambda_{j|i}} \{g_i(\lambda_{j|i})+g_j(\lambda_{i|j})-\frac{1}{(2\alpha)}||\lambda_{i|j}-\lambda_{j|i}||_2^2\}.\end{aligned} \tag{A.4}$$

From (A.4) we see that $\lambda^*_{j|i}$ and $\lambda^*_{i|j}$ are evaluated as a trade-off between maximizing $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$ and minimizing $\delta_{ij}=\lambda^*_{i|j}-\lambda^*_{j|i}$.

We consider two cases for the optimization in (A.4), one in which $C_i \cap C_j \neq \varnothing$ and one in which $C_i \cap C_j = \varnothing$.

Case $C_i \cap C_j \neq \varnothing$:

Since the intersection of C_i and C_j is a non-empty set, there exists at least an optimal point that is the global maximizer of $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$ and the global minimizer of $\frac{1}{(2\alpha)}||\lambda_{i|j}-\lambda_{j|i}||_2^2$. Therefore $\lambda^*_{j|i}$ and $\lambda^*_{i|j}$ would satisfy the following relationships:

$$\begin{aligned}\delta_{ij} &= \lambda^*_{i|j}-\lambda^*_{j|i}=0\\ g_{ij}(\lambda^*_{ij}) &= g_i(\lambda^*_{j|i})+g_j(\lambda^*_{i|j}) = g_i(\lambda^o_{j|i})+g_j(\lambda^o_{i|j}).\end{aligned} \tag{A.5}$$

Thus, $\Delta = g_{ij}(\lambda^*_{ij}) - (g_i(\lambda^*_{j|i})+g_j(\lambda^*_{i|j}))=0$.

Case $C_i \cap C_j = \varnothing$:

For the case of $C_i \cap C_j = \varnothing$, there does not exist any common optimal point as a maximizer of $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$. Therefore, the optimization in (A.4) leads to evaluation of such $\lambda^*_{j|i}$ and $\lambda^*_{i|j}$ that are not either a global maximizer of $g_i(\lambda_{j|i})$ and $g_j(\lambda_{i|j})$ nor the global minimizer of $\frac{1}{(2\alpha)}||\lambda_{i|j}-\lambda_{j|i}||_2^2$, since they are obtained as a trade off between the aforementioned maximizer and minimizer. Consequently, $\lambda^*_{j|i}$ and $\lambda^*_{i|j}$ would be two distinct values that would satisfy the following relationships:

$$\begin{aligned} \delta_{ij} &= \lambda^*_{i|j} - \lambda^*_{j|i} \neq 0 \\ g_{ij}(\lambda^*_{ij}) &< g_i(\lambda^*_{j|i}) + g_j(\lambda^*_{i|j}) < g_i(\lambda^o_{j|i}) + g_j(\lambda^o_{i|j}). \end{aligned} \tag{A.6}$$

Hence, $\Delta = g_{ij}(\lambda^*_{ij}) - (g_i(\lambda^*_{j|i}) + g_j(\lambda^*_{i|j})) < 0$.
Finally, according to (A.5) and (A.6) we can establish the following conditions:

$$\begin{aligned} \Delta = 0 \qquad &\text{if} \qquad \delta_{ij} = 0, \\ \Delta < 0 \qquad &\text{if} \qquad \delta_{ij} \neq 0. \end{aligned} \tag{A.7}$$

B

B.1 Derivation of the simple synchronous algorithm from AMM

According to (5.23) the dual variable $\lambda_{i|j}$ is updated as follows:

$$\lambda_{i|j}^{k+1} = \arg\max_{\lambda_{i|j}} \{\lambda_{i|j}^T (c_{ij} - A_{i\to j} x_i^{k+1} - A_{j\to i} x_j^k) - \frac{1}{(2\alpha)} ||\lambda_{i|j} - \lambda_{j|i}^k||_2^2\}. \quad \text{(B.1)}$$

The function on the right side of the equation (B.1), which we denote it by $h_k(\lambda_{i|j})$, is a strictly concave function with a unique maximizer. To derive the maximizer of the function, we take its gradient with respect to $\lambda_{i|j}$ as:

$$\nabla h_k(\lambda_{i|j}) = c_{ij} - A_{i\to j} x_i^{k+1} - A_{j\to i} x_j^k - \frac{1}{\alpha}(\lambda_{i|j} - \lambda_{j|i}^k). \quad \text{(B.2)}$$

By setting $\nabla h_k(\lambda_{i|j})$ to zero, $\lambda_{i|j}^{k+1}$ is evaluated as:

$$\lambda_{i|j}^{k+1} = \lambda_{j|i}^k + \alpha(c_{ij} - A_{i\to j} x_i^{k+1} - A_{j\to i} x_j^k). \quad \text{(B.3)}$$

By looking to the equation (5.22) we see that there is the need for calculation of $\sum_{j\in N(i)\backslash i} \lambda_{j|i}^{k^T} A_{i\to j}$ for updating the primal variable x_i. To simplify the aforementioned computation, we consider:

$$\beta_i^k = \sum_{j \in N(i) \backslash i} -A_{i\to j}^T \lambda_{j|i}^k, \tag{B.4}$$

as the data that is needed at node i at the k^{th} iteration.
Making use of the equation (B.3), by substituting $\lambda_{j|i}^k$ with $\lambda_{i|j}^{k-1} + \alpha(c_{ij} - A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^k)$ in equation (B.4) we have:

$$\beta_i^k = \sum_{j \in N(i) \backslash i} -A_{i\to j}^T \{\lambda_{i|j}^{k-1} + \alpha(c_{ij} - A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^k)\}, \tag{B.5}$$

we can follow the procedure and substitute $\lambda_{i|j}^{k-1}$, in (B.5), with $\lambda_{j|i}^{k-2} + \alpha(c_{ij} - A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^{k-2})$ which leads to:

$$\begin{aligned}
\beta_i^k &= \sum_{j \in N(i) \backslash i} -A_{i\to j}^T \{\lambda_{j|i}^{k-2} + \alpha(c_{ij} - A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^{k-2}) + \alpha(c_{ij} - A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^k)\} \\
&= \sum_{j \in N(i) \backslash i} -A_{i\to j}^T \lambda_{j|i}^{k-2} + \alpha \sum_{j \in N(i) \backslash i} -A_{i\to j}^T (2c_{ij} - 2A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^{k-2} - A_{j\to i}x_j^k) \\
&= \beta_i^{k-2} + \alpha \sum_{j \in N(i) \backslash i} -A_{i\to j}^T (2c_{ij} - 2A_{i\to j}x_i^{k-1} - A_{j\to i}x_j^{k-2} - A_{j\to i}x_j^k).
\end{aligned} \tag{B.6}$$

Therefore, the updates in (5.22) and (5.23) can reduce to the following updates in the synchronous case:

$$x_i^{k+1} = \arg\min_{x_i} (f_i(x_i) + \beta_i^{k^T} x_i + \sum_{j \in N(i) \backslash i} \frac{\rho}{2} ||c_{ij} - A_{i\to j}x_i - A_{j\to i}x_j^k||_2^2) \tag{B.7}$$

$$\beta_i^{k+1} = \beta_i^{k-1} + \alpha \sum_{j \in N(i) \backslash i} -A_{i\to j}^T (2c_{ij} - 2A_{i\to j}x_i^k - A_{j\to i}x_j^{k-1} - A_{j\to i}x_j^{k+1}). \tag{B.8}$$

C

C.1 Derivation of the stable BSS algorithm

The set of all $N \times N$ invertible matrices A, the parameter matrices of the blind source separation, forms a Lie group, $Gl(N)$. $Gl(N)$ is an N^2 dimensional non-linear space or manifold. In this N^2 dimensional space we can define a subspace for all matrices with equivalent property. The equivalence can be defined in terms of the directions of the column vectors of the matrices. By making use of the aforementioned property, we can define an N dimensional subspace that consists of all matrices that are only different in terms of the scale of their column vectors. We consider the matrices that forms this subspace equivalent and define a class C_A for them as:

$$C_A = \{A' | A' = A\Lambda\}, \tag{C.1}$$

where $\Lambda = \text{diag}[\lambda_{11}, \lambda_{22}, ..., \lambda_{NN}]$ is an arbitrary diagonal scaling matrix.

We refer to all those matrices that belong to C_A with A_C. Let dA_C be a direction tangent to C_A. To avoid stability problems that are related to learning scaled version of the columns of the parameter matrix A, we are interested in trajectories of learning that are orthogonal to C_A. To obtain these trajectories we require the following equality to hold

$$< dA, dA_C >_A = 0. \tag{C.2}$$

The inner product $< dA, dA_C >_A$ can be represented as

$$< dA, dA_C >_A = < A^{-1}dA, A^{-1}dA_C >_I . \tag{C.3}$$

By substituting dA_C with $Ad\Lambda$ in (C.3), it reduces to

$$\begin{aligned} < dA, dA_C >_A &= < A^{-1}dA, d\Lambda >_I \\ &= \sum_{i,j} \{A^{-1}dA\}_{ij} d\lambda_{ij} \\ &= \sum_{i=1}^{N} \{A^{-1}dA\}_{ii} d\lambda_{ii} \end{aligned} \tag{C.4}$$

where $\{A^{-1}dA\}_{ij}$ denotes the ij^{th} element of the matrix $A^{-1}dA$. Therefore, $< dA, dA_C >_A$ would be zero if $\{A^{-1}dA\}_{ii}$ be equal to zero for $i = 1, ..., N$.
From (6.5) we have:

$$A(t+1) = A(t) - \mu A(t)F[y(t)] \tag{C.5}$$

Thus, $dA = -\mu A(t)F[y(t)]$ at time index t and consequently:

$$A^{-1}dA = -\mu F[y(t)]. \tag{C.6}$$

Therefore, to satisfy the orthogonality of dA and dA_C, $\{F[y(t)]\}_{ii}$ should be zero for $i = 1, ..., N$.
Let $F[y(t)]$ to be equal to $\Lambda(t) - \psi[y(t)]y(t)^T$, where $\Lambda(t)$ is an arbitrary diagonal matrix at time index t. To satisfy the condition $\{F[y(t)]\}_{ii} = 0$, $\Lambda(t)$ should be chosen as follows:

$$\Lambda(t) = \mathrm{diag}[\mathrm{diag}[\psi[y(t)]y(t)^T]]. \tag{C.7}$$

Bibliography

[1] Abdi, H., and Williams, L. J. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics 2*, 4 (2010), 433–459.

[2] Adali, T., Kim, T., and Calhoun, V. Independent component analysis by complex nonlinearities. In *Proc. ICASSP* (2004), vol. 5, Citeseer, pp. 525–528.

[3] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A survey on sensor networks. *Communications magazine, IEEE 40*, 8 (2002), 102–114.

[4] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer networks 38*, 4 (2002), 393–422.

[5] Alavi, S. M., and Kleijn, W. B. Distributed linear blind source separation over wireless sensor networks with arbitrary connectivity patterns. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 3171–3175.

[6] Albrecht, J. R., Tuttle, C., Snoeren, A. C., and Vahdat, A. Loose synchronization for large-scale networked systems. In *USENIX Annual Technical Conference, General Track* (2006), pp. 301–314.

[7] Amari, S. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 3 (1967), 299–307.

[8] Amari, S. Natural gradient works efficiently in learning. *Neural computation 10*, 2 (1998), 251–276.

[9] Amari, S., Chen, T., and Cichoki, A. Stability analysis of adaptive blind source separation, accettato da neural networks. *Paris' ICA and BSS* (1997).

[10] Amari, S., Chen, T. P., and Cichocki, A. Nonholonomic orthogonal learning algorithms for blind source separation. *Neural computation 12*, 6 (2000), 1463–1484.

[11] Amari, S., Cichocki, A., and Yang, H. H. A new learning algorithm for blind signal separation. *Advances in neural information processing systems* (1996), 757–763.

[12] Amari, S.-I. Natural gradient learning for over-and under-complete bases in ica. *Neural Computation 11*, 8 (1999), 1875–1883.

[13] Araujo, A., and Giné, E. *The central limit theorem for real and Banach valued random variables*, vol. 431. Wiley New York, 1980.

[14] Arcangeli, A., Squartini, S., and Piazza, F. An alternative natural gradient approach for ica based learning algorithms in blind source separation. In *12th European Signal Processing Conference* (2004), IEEE, pp. 593–596.

[15] Axelsson, O. *Iterative solution methods.* Cambridge university press, 1996.

[16] Back, A. D., and Weigend, A. S. A first application of independent component analysis to extracting structure from stock returns. *International journal of neural systems 8*, 04 (1997), 473–484.

[17] Balanda, K. P., and MacGillivray, H. Kurtosis: a critical review. *The American Statistician 42*, 2 (1988), 111–119.

[18] Bartlett, M. S. *Face image analysis by unsupervised learning*, vol. 612. Springer Science & Business Media, 2012.

[19] Beck, A., and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences 2*, 1 (2009), 183–202.

[20] BELL, A. J., AND SEJNOWSKI, T. J. An information-maximization approach to blind separation and blind deconvolution. *Neural computation 7*, 6 (1995), 1129–1159.

[21] BERTSEKAS, D. P. *Nonlinear programming.* Athena scientific Belmont, 1999.

[22] BERTSEKAS, D. P., AND TSITSIKLIS, J. N. *Parallel and distributed computation: numerical methods*, vol. 23. Prentice hall Englewood Cliffs, NJ, 1989.

[23] BIANCHI, P., AND JAKUBOWICZ, J. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control 58*, 2 (2013), 391–405.

[24] BOYD, S., PARIKH, N., CHU, E., PELEATO, B., AND ECKSTEIN, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning 3*, 1 (2011), 1–122.

[25] BOYD, S., AND VANDENBERGHE, L. *Convex optimization.* Cambridge university press, 2004.

[26] BRILLOUIN, L. The negentropy principle of information. *Journal of Applied Physics 24*, 9 (1953), 1152–1163.

[27] BRUCKSTEIN, A. M., DONOHO, D. L., AND ELAD, M. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review 51*, 1 (2009), 34–81.

[28] CANDÈS, E. J., LI, X., MA, Y., AND WRIGHT, J. Robust principal component analysis? *Journal of the ACM (JACM) 58*, 3 (2011), 11.

[29] CARDOSO, J.-F. Infomax and maximum likelihood for blind source separation. *IEEE Signal Processing Letters* (1997).

[30] CARDOSO, J.-F., AND LAHELD, B. H. Equivariant adaptive source separation. *IEEE Transactions on signal processing 44*, 12 (1996), 3017–3030.

[31] Chandrasekaran, V., Parrilo, P. A., and Willsky, A. S. Latent variable graphical model selection via convex optimization. In *48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* (2010), IEEE, pp. 1610–1613.

[32] Chen, C., He, B., Ye, Y., and Yuan, X. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming 155*, 1-2 (2016), 57–79.

[33] Chen, S., and Donoho, D. Basis pursuit. In *Conference Record of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers* (1994), vol. 1, IEEE, pp. 41–44.

[34] Christopher, M. B. Pattern recognition and machine learning. *Company New York 16*, 4 (2006), 049901.

[35] Cichocki, A., and Amari, S.-i. *Adaptive blind signal and image processing: learning algorithms and applications*, vol. 1. John Wiley & Sons, 2002.

[36] Cichocki, A., Unbehauen, R., Moszczynski, L., and Rummert, E. A new on-line adaptive learning algorithm for blind separation of source signals. In *Proc. ISANN* (1994), vol. 94, pp. 406–411.

[37] Comon, P. Independent component analysis, a new concept? *Signal processing 36*, 3 (1994), 287–314.

[38] Comon, P., and Jutten, C. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[39] Comon, P., Jutten, C., and Herault, J. Blind separation of sources, part ii: Problems statement. *Signal processing 24*, 1 (1991), 11–20.

[40] Cover, T. M., and Thomas, J. A. *Elements of information theory.* John Wiley & Sons, 2012.

[41] DeCarlo, L. T. On the meaning and use of kurtosis. *Psychological methods 2*, 3 (1997), 292.

[42] Deng, W., Lai, M. J., Peng, Z., and Yin, W. Parallel multi-block ADMM with $O(1/k)$ convergence. *arXiv preprint arXiv:1312.3040* (2013).

[43] Deng, W., and Yin, W. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing* (2012), 1–28.

[44] Dmochowski, J. P., Liu, Z., and Chou, P. A. Blind source separation in a distributed microphone meeting environment for improved teleconferencing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2008), IEEE, pp. 89–92.

[45] Du, H., Qi, H., and Wang, X. A parallel independent component analysis algorithm. In *12th International Conference on Parallel and Distributed Systems (ICPADS)* (2006), vol. 1, IEEE, pp. 151–160.

[46] Duchi, J. C., Agarwal, A., and Wainwright, M. J. Dual averaging for distributed optimization: convergence analysis and network scaling. *IEEE Transactions on Automatic control 57*, 3 (2012), 592–606.

[47] Eckstein, J., and Bertsekas, D. P. On the douglasâĂŤrachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming 55*, 1-3 (1992), 293–318.

[48] Eckstein, J., and Yao, W. Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports 32* (2012).

[49] Everett III, H. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research 11*, 3 (1963), 399–417.

[50] Facchinei, F., and Pang, J.-S. *Finite-dimensional variational inequalities and complementarity problems.* Springer Science & Business Media, 2007.

[51] Forero, P., Cano, A., Giannakis, G. B., et al. Distributed clustering using wireless sensor networks. *IEEE Journal of Selected Topics in Signal Processing 5*, 4 (2011), 707–724.

[52] Freund, R. M. Penalty and barrier methods for constrained optimization. *Lecture Notes, Massachusetts Institute of Technology* (2004).

[53] Fujieda, M., Murakami, T., and Ishida, Y. An approach to solving a permutation problem of frequency domain independent component analysis for blind source separation of speech signals. *International Journal of Biological and Life Sciences 1* (2005), 4.

[54] Gabay, D. Chapter ix applications of the method of multipliers to variational inequalities. *Studies in mathematics and its applications 15* (1983), 299–331.

[55] Gabay, D., and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications 2*, 1 (1976), 17–40.

[56] Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control 60*, 3 (2015), 644–658.

[57] Girolami, M. A variational method for learning sparse and overcomplete representations. *Neural computation 13*, 11 (2001), 2517–2532.

[58] Glowinski, R. Numerical methods for nonlinear variational methods, 1984.

[59] Godsil, C., and Royle, G. Algebraic graph theory. *Graduate Texts in Mathematics 207* (2001).

[60] Goldfarb, D., Ma, S., and Scheinberg, K. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming 141*, 1-2 (2013), 349–382.

[61] Goldstein, A. A. Convex programming in hilbert space. *Bulletin of the American Mathematical Society 70*, 5 (1964), 709–710.

[62] Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences 7*, 3 (2014), 1588–1623.

[63] Gowrishankar, S., Basavaraju, T. G., Manjaiah, D. H., and Sarkar, S. K. Issues in wireless sensor networks. In *Proceedings of the World Congress on Engineering* (2008), vol. 1, pp. 978–988.

[64] Grubbs, F. E. Procedures for detecting outlying observations in samples. *Technometrics 11*, 1 (1969), 1–21.

[65] Han, D., Yuan, X., and Zhang, W. An augmented lagrangian based parallel splitting method for separable convex minimization with applications to image processing. *Mathematics of Computation 83*, 289 (2014), 2263–2291.

[66] Harman, H. H. *Modern factor analysis.* University of Chicago Press, 1976.

[67] Haykin, S., and Network, N. A comprehensive foundation. *Neural Networks 2*, 2004 (2004).

[68] Haykin, S., and Widrow, B. *Least-mean-square adaptive filters*, vol. 31. John Wiley & Sons, 2003.

[69] Haykin, S. S. *Unsupervised adaptive filtering: Blind source separation*, vol. 1. Wiley-Interscience, 2000.

[70] Haykin, S. S. *Adaptive filter theory.* Pearson Education India, 2008.

[71] He, B., Hou, L., and Yuan, X. On full jacobian decomposition of the augmented lagrangian method for separable convex programming. *SIAM Journal on Optimization 25*, 4 (2015), 2274–2312.

[72] He, B., Tao, M., and Yuan, X. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization 22*, 2 (2012), 313–340.

[73] He, B., and Yuan, X. On the $o(1/n)$ convergence rate of the douglas-rachford alternating direction method. *SIAM Journal on Numerical Analysis 50*, 2 (2012), 700–709.

[74] He, B., and Yuan, X. On the direct extension of admm for multi-block separable convex program-ming and beyond: from variational inequality perspective. *Manuscript, http://âĂŃ www.âĂŃ optimizationon-liâĂŃ ne.âĂŃ org/âĂŃ DB_ âĂŃ HTML 3* (2014), 4293.

[75] He, B.-S. Parallel splitting augmented lagrangian methods for monotone structured variational inequalities. *Computational Optimization and Applications 42*, 2 (2009), 195–212.

[76] Hestenes, M. R. Multiplier and gradient methods. *Journal of optimization theory and applications 4*, 5 (1969), 303–320.

[77] Heusdens, R., Zhang, G., Hendriks, R. C., Zeng, Y., and Kleijn, W. B. Distributed mvdr beamforming for (wireless) microphone networks using message passing. In *International Workshop on Acoustic Signal Enhancement; Proceedings of IWAENC* (2012), VDE, pp. 1–4.

[78] Hild II, K. E., Erdogmus, D., and Principe, J. C. On-line minimum mutual information method for time-varying blind source separation. In *Proceedings of the 3 rd International Symposium on Independent Component Analysis and Signal Separation, ICA 2001* (2001), pp. 126–131.

[79] Hioka, Y., and Kleijn, W. B. Distributed blind source separation with an application to audio signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), IEEE, pp. 233–236.

[80] Hiriart-Urruty, J.-B., and Martinez-Legaz, J.-E. New formulas for the legendre–fenchel transform. *Journal of mathematical analysis and applications 288*, 2 (2003), 544–555.

[81] Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology 24*, 6 (1933), 417.

[82] Huber, P. J. Projection pursuit. *The annals of Statistics* (1985), 435–475.

[83] Hyvarinen, A. New approximations of differential entropy for independent component analysis and projection pursuit. *Advances in neural information processing systems 10*, 2 (1998), 273–279.

[84] Hyvärinen, A. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks 10*, 3 (1999), 626–634.

[85] Hyvärinen, A., Karhunen, J., and Oja, E. *Independent component analysis*, vol. 46. John Wiley & Sons, 2004.

[86] Hyvärinen, A., and Oja, E. Independent component analysis: algorithms and applications. *Neural networks 13*, 4 (2000), 411–430.

[87] Hyviirinen, A., Karhunen, J., and Oja, E. Independent component analysis. *Wileyand Sons* (2001).

[88] Ikram, M. Z., and Morgan, D. R. A beamforming approach to permutation alignment for multichannel frequency-domain blind speech separation. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2002), vol. 1, IEEE, pp. I–881.

[89] Ikram, M. Z., and Morgan, D. R. Permutation inconsistency in blind speech separation: investigation and solutions. *IEEE Transactions on Speech and Audio Processing 13*, 1 (2005), 1–13.

[90] Iutzeler, F., Bianchi, P., Ciblat, P., and Hachem, W. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd Annual Conference on Decision and Control (CDC)* (2013), IEEE, pp. 3671–3676.

[91] Jiang, Z., and Yuan, X. New parallel descent-like method for solving a class of variational inequalities. *Journal of optimization theory and applications 145*, 2 (2010), 311–323.

[92] JOHNSON, J. K., BICKSON, D., AND DOLEV, D. Fixing convergence of gaussian belief propagation. In *IEEE International Symposium on Information Theory (ISIT)* (2009), IEEE, pp. 1674–1678.

[93] JOLLIFFE, I. *Principal component analysis.* Wiley Online Library, 2002.

[94] JUTTEN, C., AND HERAULT, J. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing 24*, 1 (1991), 1–10.

[95] KADKHODAIE, M., CHRISTAKOPOULOU, K., SANJABI, M., AND BANERJEE, A. Accelerated alternating direction method of multipliers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), ACM, pp. 497–506.

[96] KARUSH, W. *Minima of functions of several variables with inequalities as side constraints.* PhD thesis, MasterâĂŹs thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[97] KJELDSEN, T. H. A contextualized historical analysis of the kuhn–tucker theorem in nonlinear programming: The impact of world war ii. *Historia mathematica 27*, 4 (2000), 331–361.

[98] KLEIJN, W. Distributed blind source separation, Oct. 23 2012. US Patent 8,295,762.

[99] KLEMM, M., HAUEISEN, J., AND IVANOVA, G. Independent component analysis: comparison of algorithms for the investigation of surface electrical brain activity. *Medical & biological engineering & computing 47*, 4 (2009), 413–423.

[100] KNOTT, M., AND BARTHOLOMEW, D. J. *Latent variable models and factor analysis.* No. 7. Edward Arnold, 1999.

[101] KOEHLER, T.-W. L. B.-U. Blind source separation of nonlinear mixing models. *Neural Networks for Signal Processing VII* (1997), 406.

[102] Koldovskỳ, Z., and Tichavskỳ, P. Comparison of independent component and independent subspace analysis algorithms. In *Signal Processing Conference, 2009 17th European* (2009), IEEE, pp. 1447–1451.

[103] Koller, D., and Friedman, N. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

[104] Kuhn, H., and Tucker, A. Proceedings of 2nd berkeley symposium, 1951.

[105] Lappalainen, H., and Honkela, A. Bayesian non-linear independent component analysis by multi-layer perceptrons. In *Advances in independent component analysis.* Springer, 2000, pp. 93–121.

[106] LeBlanc, J. P., and De Leon, P. L. Speech separation by kurtosis maximization. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on* (1998), vol. 2, IEEE, pp. 1029–1032.

[107] Lee, S. Blind source separation and independent component analysis: A review. *Neural Information Processing- Letters and Reviews 6*, 1 (2005).

[108] Lee, T. *Independent component analysis: theory and applications.* Kluwer Academic Publishers, 1997.

[109] Lee, T., Lewicki, M. S., Girolami, M., and Sejnowski, T. J. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters 6*, 4 (1999), 87–90.

[110] Lee, T.-W., Girolami, M., and Sejnowski, T. J. Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural computation 11*, 2 (1999), 417–441.

[111] Lesser, V., Ortiz Jr, C. L., and Tambe, M. *Distributed sensor networks: A multiagent perspective*, vol. 9. Springer Science & Business Media, 2012.

[112] Levitin, E. S., and Polyak, B. T. Constrained minimization methods. *USSR Computational mathematics and mathematical physics 6*, 5 (1966), 1–50.

[113] Lewicki, M., and Sejnowski, T. Learning overcomplete representations. *Neural computation 12*, 2 (2000), 337–365.

[114] Li, H., and Adali, T. A class of complex ica algorithms based on the kurtosis cost function. *Neural Networks, IEEE Transactions on 19*, 3 (2008), 408–420.

[115] Lin, T., Ma, S., and Zhang, S. On the sublinear convergence rate of multi-block admm. *Journal of the Operations Research Society of China 3*, 3 (2015), 251–274.

[116] Lions, P.-L., and Mercier, B. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis 16*, 6 (1979), 964–979.

[117] Liu, C., and Wechsler, H. Independent component analysis of gabor features for face recognition. *IEEE Transactions on Neural Networks 14*, 4 (2003), 919–928.

[118] Liu, W., and Mandic, D. P. A normalised kurtosis-based algorithm for blind source extraction from noisy measurements. *Signal Processing 86*, 7 (2006), 1580–1585.

[119] Lu, C.-J., Lee, T.-S., and Chiu, C.-C. Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems 47*, 2 (2009), 115–125.

[120] Luenberger, D. G. *Introduction to linear and nonlinear programming*, vol. 28. Addison-Wesley Reading, MA, 1973.

[121] Maddala, G. S., and Lahiri, K. *Introduction to econometrics*, vol. 2. Macmillan New York, 1992.

[122] Makeig, S., Bell, A. J., Jung, T.-P., Sejnowski, T. J., et al. Independent component analysis of electroencephalographic data. *Advances in neural information processing systems* (1996), 145–151.

[123] Margaris, A. I., and Diamantaras, K. I. A parallel implementation of the natural gradient BSS method using MPI. In *2nd Int. Conference on Experiments/Process/System Modeling/Simulation & Optimization* (2007).

[124] Mateos, G., Bazerque, J. A., and Giannakis, G. B. Distributed sparse linear regression. *IEEE Transactions on Signal Processing 58*, 10 (2010), 5262–5276.

[125] McDonald, R., Hall, K., and Mann, G. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010), Association for Computational Linguistics, pp. 456–464.

[126] Merris, R. Laplacian matrices of graphs: a survey. *Linear algebra and its applications 197* (1994), 143–176.

[127] Miele, A., Moseley, P., Levy, A., and Coggins, G. On the method of multipliers for mathematical programming problems. *Journal of Optimization Theory and Applications 10*, 1 (1972), 1–33.

[128] Mitianoudis, N., and Davies, M. E. Audio source separation of convolutive mixtures. *IEEE Transactions on Speech and Audio Processing 11*, 5 (2003), 489–497.

[129] Moallemi, C. C., and Roy, B. V. Distributed optimization in adaptive networks. In *Advances in Neural Information Processing Systems* (2003).

[130] Moreau, E., and Maachi, O. High-order contrasts for self-adaptive source separation. *International Journal of Adaptive Control and Signal Processing 10*, 1 (1996), 19–46.

[131] Mota, J. F., Xavier, J. M., Aguiar, P. M., and Puschel, M. D-admm: A communication-efficient distributed algorithm for separable op-

timization. *IEEE Transactions on Signal Processing 61*, 10 (2013), 2718–2723.

[132] Murphy, K. P., Weiss, Y., and Jordan, M. I. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence* (1999), Morgan Kaufmann Publishers Inc., pp. 467–475.

[133] Murray, R. O. S., and Murray, R. M. Consensus protocols for networks of dynamic agents. In *Proceedings of the American Controls Conference* (2003).

[134] Mutihac, R., and Mutihac, R. C. A comparative study of independent component analysis algorithms for electroencephalography. *Romanian reports in physics 59*, 3 (2007), 827–853.

[135] Nadal, J.-P., and Parga, N. Nonlinear neurons in the low-noise limit: a factorial code maximizes information transfer. *Network: Computation in neural systems 5*, 4 (1994), 565–581.

[136] Nedić, A., and Ozdaglar, A. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control 54*, 1 (2009), 48–61.

[137] Nedic, A., and Ozdaglar, A. Cooperative distributed multi-agent optimization. *Convex Optimization in Signal Processing and Communications 340* (2010).

[138] Nesterov, Y. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. In *Soviet Mathematics Doklady* (1983), vol. 27, pp. 372–376.

[139] Novey, M., and Adali, T. Complex ica by negentropy maximization. *IEEE Transactions on Neural Networks 19*, 4 (2008), 596–609.

[140] Nuzillard, D., and Bijaoui, A. Blind source separation and analysis of multispectral astronomical images. *Astronomy and Astrophysics Supplement Series 147*, 1 (2000), 129–138.

[141] Ouyang, Y., Chen, Y., Lan, G., and Pasiliao Jr, E. An accelerated linearized alternating direction method of multipliers. *SIAM Journal on Imaging Sciences 8*, 1 (2015), 644–681.

[142] Pahm, D., Garrat, P., and Jutten, C. Separation of a mixture of independent sources through a ml approach. In *Proc. European Signal Processing Conf* (1992), p. 771.

[143] Papadias, C. B. Globally convergent blind source separation based on a multiuser kurtosis maximization criterion. *IEEE Transactions on Signal Processing 48*, 12 (2000), 3508–3519.

[144] Papoulis, A., and Pillai, S. U. *Probability, random variables, and stochastic processes.* Tata McGraw-Hill Education, 2002.

[145] Parikh, N., and Boyd, S. Block splitting for distributed optimization. *Mathematical Programming Computation 6*, 1 (2014), 77–102.

[146] Pearson, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 2*, 11 (1901), 559–572.

[147] Pedersen, M. S., Larsen, J., Kjems, U., and Parra, L. C. A survey of convolutive blind source separation methods. *Multichannel Speech Processing Handbook* (2007), 1065–1084.

[148] Peng, Y., Ganesh, A., Wright, J., Xu, W., and Ma, Y. Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 11 (2012), 2233–2246.

[149] Perlmutter, B., and Parra, L. C. Maximum likelihood blind source separation: A context-sensitive generalization of ica. In *Neural Information Processing Systems* (1997), vol. 9, pp. 613–619.

[150] Pham, D.-T. Fast algorithms for mutual information based independent component analysis. *Signal Processing, IEEE Transactions on 52*, 10 (2004), 2690–2700.

[151] Pham, D. T., and Garat, P. Blind separation of mixture of independent sources through a maximum likelihood approach. In *In Proc. EUSIPCO* (1997), Citeseer.

[152] Powell, M. J. *A method for non-linear constraints in minimization problems.* UKAEA, 1967.

[153] Qin, Z., Goldfarb, D., and Ma, S. An alternating direction method for total variation denoising. *Optimization Methods and Software 30*, 3 (2015), 594–615.

[154] Rabbat, M., and Nowak, R. Distributed optimization in sensor networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks* (2004), ACM, pp. 20–27.

[155] Ram, S. S., Nedić, A., and Veeravalli, V. V. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications 147*, 3 (2010), 516–545.

[156] Rice, J. *Mathematical statistics and data analysis.* Nelson Education, 2006.

[157] Robledo-Arnuncio, E., and Juang, B. H. Blind source separation of acoustic mixtures with distributed microphones. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2007), vol. 3, IEEE, pp. III 949 – III 952.

[158] Rockafellar, R. Conjugates and legendre transforms of convex functions. *Canad. J. Math 19* (1967), 200–205.

[159] Rojas, F., Rojas, I., Clemente, R., and Puntonet, C. Nonlinear blind source separation using genetic algorithms. In *Proceedings of International Conference on Independent Component Analysis* (2001).

[160] Rudin, L. I., Osher, S., and Fatemi, E. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*, 1 (1992), 259–268.

[161] Rudin, W. Principles of mathematical analysis (international series in pure & applied mathematics).

[162] Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. *Parallel distributed processing*, vol. 1. IEEE, 1988.

[163] Ruppert, D. What is kurtosis? an influence function approach. *The American Statistician 41*, 1 (1987), 1–5.

[164] Saad, Y. Iterative methods for sparse linear systems pws. *New York* (1996).

[165] Sawada, H., Mukai, R., Araki, S., and Makino, S. A robust and precise method for solving the permutation problem of frequency-domain blind source separation. *IEEE Transactions on Speech and Audio Processing 12*, 5 (2004), 530–538.

[166] Sayed, A. Adaptive filters. hoboken. *NJ: John Wiley & Sons 10* (2008), 9780470374122.

[167] Sayed, A. H., Tu, S.-Y., Chen, J., Zhao, X., and Towfic, Z. J. Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior. *IEEE Signal Processing Magazine 30*, 3 (2013), 155–171.

[168] Schizas, I. D., Ribeiro, A., and Giannakis, G. B. Consensus in ad hoc wsns with noisy linksâĂŤpart i: Distributed estimation of deterministic signals. *IEEE Transactions on Signal Processing 56*, 1 (2008), 350–364.

[169] Shental, O., Siegel, P. H., Wolf, J. K., Bickson, D., and Dolev, D. Gaussian belief propagation solver for systems of linear equations. In *IEEE International Symposium on Information Theory (ISIT)* (2008), IEEE, pp. 1863–1867.

[170] Shor, N. Z. *Minimization methods for non-differentiable functions*, vol. 3. Springer Science & Business Media, 2012.

[171] Smaragdis, P. J. *Information theoretic approaches to source separation.* PhD thesis, Massachusetts Institute of Technology, 1997.

[172] Smith, A. E., and Coit, D. W. Penalty functions. *Handbook on Evolutionary Computation, pages C 5* (1997), 1–6.

[173] Sorouchyari, E. Blind separation of sources, part iii: Stability analysis. *Signal processing 24*, 1 (1991), 21–29.

[174] Squartini, S., Piazza, F., and Shawker, A. New riemannian metrics for improvement of convergence speed in ica based learning algorithms. In *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on* (2005), IEEE, pp. 3603–3606.

[175] Sternberg, R. J. *Metaphors of mind: Conceptions of the nature of intelligence.* Cambridge University Press, 1990.

[176] Tao, M., and Yuan, X. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization 21*, 1 (2011), 57–81.

[177] Thompson, B. *Exploratory and confirmatory factor analysis: Understanding concepts and applications.* American Psychological Association, 2004.

[178] Toyama, K., and Plumbley, M. D. Using phase linearity in frequency-domain ica to tackle the permutation problem. In *International Conference on Acoustics, Speech and Signal Processing, 2009. ICASSP 2009.* (2009), IEEE, pp. 3165–3168.

[179] Uzawa, H. Market mechanisms and mathematical programming. *Econometrica: Journal of the Econometric Society* (1960), 872–881.

[180] Uzawa, H. Walras' tatonnement in the theory of exchange. *The Review of Economic Studies 27*, 3 (1960), 182–194.

[181] Vapnyarskii, I. Lagrange multipliers. *Hazewinkel, Michiel, Encyclopedia of Mathematics, Springer, ISBN* (2001), 978–1.

[182] Vieira, M. A. M., Coelho Jr, C. N., da Silva, D., and da Mata, J. M. Survey on wireless sensor network devices. In *Conference on Emerg-*

ing Technologies and Factory Automation (ETFA) (2003), vol. 1, IEEE, pp. 537–544.

[183] VIGÁRIO, R., SÄRELÄ, J., JOUSMIKI, V., HÄMÄLÄINEN, M., AND OJA, E. Independent component approach to the analysis of eeg and meg recordings. *IEEE Transactions on Biomedical Engineering 47*, 5 (2000), 589–593.

[184] VINCENT, E., FÉVOTTE, C., GRIBONVAL, R., BENAROYA, L., RODET, X., RÖBEL, A., LE CARPENTIER, E., AND BIMBOT, F. A tentative typology of audio source separation tasks. In *4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA)* (2003), pp. 715–720.

[185] WAINWRIGHT, M. J., AND JORDAN, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning 1*, 1-2 (2008), 1–305.

[186] WANG, Q., HEMPSTEAD, M., AND YANG, W. A realistic power consumption model for wireless sensor network devices. In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)* (2006), vol. 1, IEEE, pp. 286–295.

[187] WANG, X., HONG, M., MA, S., AND LUO, Z.-Q. Solving multiple-block separable convex minimization problems using two-block alternating direction method of multipliers. *arXiv preprint arXiv:1308.5294* (2013).

[188] WEI, E., AND OZDAGLAR, A. On the $o(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In *Global Conference on Signal and Information Processing (GlobalSIP)* (2013), IEEE, pp. 551–554.

[189] WIDROW, B. A statistical theory of adaptation. *Adaptive Control Systems (eds.: Caruthers, FP and Levenstein, H.). Pergamon Press, Oxford 196* (1963), 97–122.

[190] WOLD, S., ESBENSEN, K., AND GELADI, P. Principal component analysis. *Chemometrics and intelligent laboratory systems 2*, 1 (1987), 37–52.

[191] Xiao, L., and Boyd, S. Fast linear iterations for distributed averaging. *Systems & Control Letters 53*, 1 (2004), 65–78.

[192] Xiao, L., Boyd, S., and Kim, S.-J. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing 67*, 1 (2007), 33–46.

[193] Yalcin, I., and Amemiya, Y. Nonlinear factor analysis as a statistical method. *Statistical science* (2001), 275–294.

[194] Yang, H. H., and Amari, S.-i. Adaptive online learning algorithms for blind separation: maximum entropy and minimum mutual information. *Neural computation 9*, 7 (1997), 1457–1482.

[195] Yang, H. H., Amari, S.-I., and Cichocki, A. Information theoretic approach to blind separation of sources in non-linear mixture. *Signal Processing 64*, 3 (1998), 291–300.

[196] Yick, J., Mukherjee, B., and Ghosal, D. Wireless sensor network survey. *Computer networks 52*, 12 (2008), 2292–2330.

[197] Zakon, E. *Mathematical analysis.* The Trillia Group, 2004.

[198] Zarzoso, V., and Nandi, A. Blind source separation. In *Blind Estimation Using Higher-Order Statistics.* Springer, 1999, pp. 167–252.

[199] Zarzoso, V., and Nandi, A. K. Adaptive blind source separation for virtually any source probability density function. *IEEE Transactions on signal processing 48*, 2 (2000), 477–488.

[200] Zhang, G., and Heusdens, R. Generalized linear coordinate-descent message-passing for convex optimization. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012), IEEE, pp. 2009–2012.

[201] Zhang, G., and Heusdens, R. Linear coordinate-descent message passing for quadratic optimization. *Neural computation 24*, 12 (2012), 3340–3370.

[202] Zhang, G., and Heusdens, R. Bi-alternating direction method of multipliers over graphs. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), IEEE.

[203] Zhang, G., Heusdens, R., and Kleijn, W. B. Large scale lp decoding with low complexity. *IEEE Communications Letters 17*, 11 (2013), 2152–2155.

[204] Zhang, G., Heusdens, R., and Kleijn, W. B. On the convergence rate of the bi-alternating direction method of multipliers. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), IEEE, pp. 3869–3873.

[205] Zhang, L.-Q., Cichocki, A., and Amari, S. Natural gradient algorithm for blind separation of overdetermined mixture with additive noise. *IEEE Signal Processing Letters 6*, 11 (1999), 293–295.

[206] Zhang, R., and Kwok, J. Asynchronous distributed admm for consensus optimization. In *Proceedings of the 31st International Conference on Machine Learning (ICML)* (2014), pp. 1701–1709.

[207] Zhao, X., and Sayed, A. H. Performance limits for distributed estimation over lms adaptive networks. *IEEE Transactions on Signal Processing 60*, 10 (2012), 5107–5124.

[208] Zhou, Z., Li, X., Wright, J., Candes, E., and Ma, Y. Stable principal component pursuit. In *IEEE International Symposium on Information Theory Proceedings (ISIT)* (2010), IEEE, pp. 1518–1522.

[209] Zhu, H., Cano, A., and Giannakis, G. B. Distributed consensus-based demodulation: algorithms and error analysis. *IEEE Transactions on Wireless Communications 9*, 6 (2010), 2044–2054.

[210] Zhu, X.-L., Zhang, X.-D., Ding, Z.-Z., and Jia, Y. Adaptive nonlinear pca algorithms for blind source separation without prewhitening. *IEEE Transactions on Circuits and Systems I: Regular Papers 53*, 3 (2006), 745–753.