Technical University of Crete
School of Electrical and Computer Engineering

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Clustering of Inference Algorithms in Communication Networks

by Emmanouil Kariotakis

A Thesis Submitted in Partial Fulfillment of the Requirements for the Diploma Degree of

Electrical and Computer Engineering

September, 2022

Thesis Committee
Professor Aggelos Bletsas, *Thesis Supervisor*
Professor Michail Zervakis
Professor George N. Karystinos

Abstract

This work offers an algorithmic framework for in-network inference, using message passing among ambiently powered wireless sensor network (WSN) terminals. The stochastic nature of ambient energy harvesting dictates intermittent operation of each WSN terminal and as such, the message passing inference algorithms should be robust to asynchronous operation. A version of Gaussian Belief Algorithm (GBP) is described, which can be reduced to an affine fixed point (AFP) problem, used to solve linear systems of equations. To achieve this, we have to cluster the Probabilistic Graphical Model (PGM) behind GBP, in order to map it to the WSN terminals. We propose two different clustering approaches, namely edge and node clustering. For the first approach, we explain the reasons why a previous method does not produce the expected results and we offer another method, which performs better. We also explain limitations of edge-based clustering. On the other hand, node clustering has a clear metric for performance, which is relevant to the number of edges connecting the different clusters. For this approach, we utilize three different clustering algorithms, the k-means, the spectral clustering and an autonomous, in-network clustering algorithm. Furthermore, we show in both theory and simulation that there is strong connection between spectral radius and the convergence rate of AFP problems with probabilistic asynchronous scheduling. The latter corroborates known theory for synchronous scheduling. Interestingly, it is shown through simulations that different clustering offers similar convergence rate, when probabilistic asynchronous scheduling is utilized with carefully selected probabilities that accelerate convergence rate in the mean sense. Finally, we show an existing distinction between convergence rate and energy consumption of the network and we present experimental results comparing the different clustering methods. In most cases, spectral clustering outperforms the rest, with reduced energy consumption (by a factor of 2 compared to k-means in specific cases).

Acknowledgements

After 5 wonderful years at the Technical University of Crete, a hard yet exciting journey has come to an end. I cannot but express my gratitude towards the people that supported me throughout these years and made everything I achieved possible.

First and foremost, I would like to thank my supervisor, Prof. Aggelos Bletsas, for giving me the opportunity to conduct research into a very fascinating scientific field. I am also grateful to the other two members of my committee. I would like to thank Prof. George Karystinos for his amazing courses and his mentoring and valuable advice. I am also thankful to prof. Michail Zervakis for his support and understanding when I had to make difficult decisions. In addition, I would like to thank Prof. Athanasios Liavas for his exciting courses though which I set a strong mathematical basis. Finally, I would like to thanks all my colleagues from Bletsas Group for their great help and advice.

However, I could not have undertaken this journey without my family. My family's constant and unconditional support through every decision I made is what kept me going all these years. Thank you for always being there for me and encouraging me to reach my goals, no matter how difficult they might seem at the time! Last but not least, I cannot thank enough my closest friends. You were by my side in both good and bad times, for all these years, and we formed memories that I will never forget!

-Emmanouil Kariotakis, September 2022

Contents

Chapter 1

Introduction

Recent advances on powerful message passing algorithms (e.g. sum-product, max-product), also known as belief propagation [1, 2, 3], have offered concrete examples on how decision making and inference can be facilitated through communication at carefully crafted graphs. These algorithms have been the focus of much research; multiple extensions have been proposed and have been applied successfully to a variety of domains.

More importantly, recent advances on backscatter (or simple scatter) radio sensor networks, have demonstrated feasibility of μWatt power for low-cost, joint sensing and wireless networking [4, 5, 6, 7, 8, 9, 10, 11]; all is needed at the transmitter side are a radio frequency (RF) transistor, an antenna and a low-cost microcontroller unit (MCU). As described in previous work [12, 13], it is possible to build ultra-low power wireless sensor networks (WSN) that have no central processing unit in order to make autonomous, in-network decisions, solely powered by the environment. This is possible due to the network's *asynchronous* scheduling [12, 13].

Message passing algorithms exploit the factorization of the joint probability distribution (for discrete variables) or density function (for continuous variables) to a product of factors and the corresponding encoding of conditional (in)dependencies between variables into a carefully constructed probabilistic graphical model (PGM). The sum-product (max-product) message passing algorithm runs in these PGMs and offers the marginal densities (maximum aposteriori probability values) of the random variables.

Our goal is to take advantage of the WSNs' and the algorithms' distributed nature and carefully map the nodes of the PGMs used in such algorithms to the WSN terminals, in order to minimize the required computation and communication load across different WSN terminals or minimize the convergence time, and exploit the power of the aforementioned algorithms and make in-network decisions. More specifically, we utilize *Gaussian Belief Propagation* algorithm [14] in order to solve linear systems of equations. The matrices that are used in our experiments are provided in Appendix A.

1.1 WSN as an Inference Platform

In this section we are going to formulate the aforementioned mapping. At the end of the chapter there is a detailed explanation of the used notation.

Consider an ambiently powered WSN with N (physical) terminals. In addition let $\mathbf{x}^{(0)} \in \mathbb{R}^n$; our goal is to utilize the network in order to calculate the update:

$$\mathbf{x}^{(l)} = f\left(\mathbf{x}^{(l-1)}\right) = \left[f_1\left(\mathbf{x}_{\mathcal{J}_1}^{(l-1)}\right) \dots f_n\left(\mathbf{x}_{\mathcal{J}_n}^{(l-1)}\right)\right], \quad l = 1, 2, \dots, \tag{1.1}$$

where $\mathbf{x}_{\mathcal{J}_i}^{(l-1)}$ denotes a subset of elements $\left\{x_k^{(l-1)}\right\}$ of $\mathbf{x}^{(l-1)}$, according to the set of indices $\mathcal{J}_i$, $f : \mathbb{R}^n \to \mathbb{R}^n$ is a real mapping and $f_k : \mathbb{R}^{|\mathcal{J}_k|} \to \mathbb{R}$ is that real valued function the is used to update element $x_k^{(l-1)}$. On top of that, assume that at each iteration, WSN terminal i is responsible for updating a unique subset of the elements $\left\{x_k^{(l-1)}\right\}$ of $\mathbf{x}^{(l-1)}$, denoted by $\mathbf{x}_{\mathcal{I}_i}^{(l-1)}$, where $\bigcap_{i=1}^N \mathcal{I}_i = \emptyset$ and $\bigcup_{i=1}^N \mathcal{I}_i = \{1, \dots, n\}$, utilizing the appropriate subset of functions $\{f_j(\cdot)\}$, $j \in \mathcal{I}_i$. In other words, each WSN terminal is responsible for updating a subset of the variables, all variables are allocated to specific WSN terminals and no variable is allocated to more than one WSN terminal. Given that each function $f_j(\cdot)$ might require variables that are allocated to different WSN terminals, communication between the WSN nodes is required; before any computations, the required values must firstly be transmitted between the WSN nodes as messages. An example of such a mapping is presented in Figure 1.1, where we can see the Graphical Model, the Wireless Sensor Network and the final mapping that has been done.

However, such message passing may fail, simply because the WSN terminal does not have sufficient energy; thus the necessary message passing is interrupted probabilistically. As a result, some subsets of $\mathbf{x}^{(l-1)}$ may not be updated at iteration (l); this is what we refer to as *asynchrony*. Considering this *asynchronous* operation, messages between variable or factor nodes, that belong to different WSN terminals, may fail to be transmitted or received due to energy outage of a WSN terminal or because of a failure in transmission between WSN terminals. Let p_i^{out} denote the probability of terminal i being in energy outage, $p_{i,j}^{\text{trans}}$ denote the probability of successful transmission between terminals i and j and finally, $p_{i,j}^{\text{comm}}$ the probability of successful communication between i and j, which is $p_{i,j}^{\text{comm}} \triangleq p_{j,i}^{\text{comm}} = (1 - p_i^{\text{out}})(1 - p_j^{\text{out}})p_{i,j}^{\text{trans}}$. For simplicity, we assume that messages between WSN terminals fail to communicate only due to energy outage, i.e. $p_{i,j}^{\text{trans}} = 1$, hence,

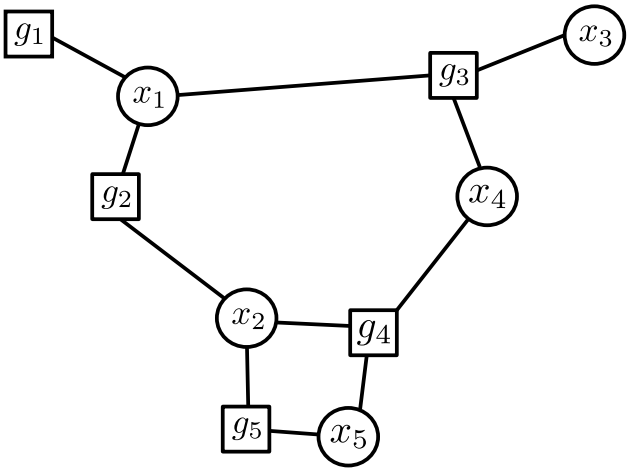
$$p_{i,j}^{\text{comm}} \triangleq p_{j,i}^{\text{comm}} = (1 - p_i^{\text{out}})(1 - p_j^{\text{out}}). \tag{1.2}$$

We also assume that messages $m_{g_j \to x_i}$ are stored in node variables x_i and that the probability of a messages being sent from g_j to x_i equals to the probability from x_i to g_j, i.e. $p_{g_j \to x_i} = p_{x_i \to g_j}$.

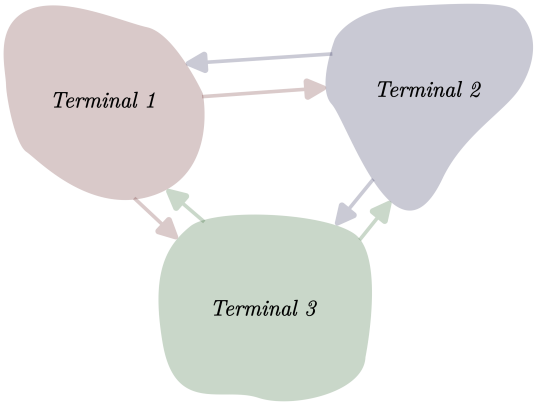
Thesis Outline

In Chapter 2 we provide the basic algorithm that we use throughout this work and some convergence conditions for it, in Chapter 3 we present some clustering methods that are used in order to

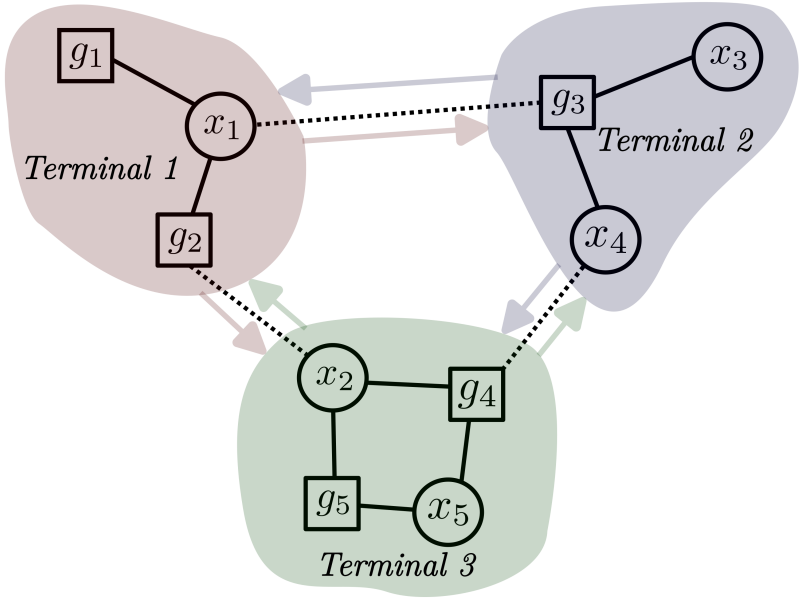
run this algorithm in communication networks. In Chapter 4 we state two optimization problems, on which we provide solutions, and we present experimental results of those problems. Finally, in Chapter 5 we state the main contributions and propose future directions of our work.



(a) An example of a Probabilistic Graphical Model that we want to map.



(b) A Wireless Sensor Network with 3 physical Terminals.



(c) The Network after the mapping is performed.

Figure 1.1: An example of mapping a PGM to a WSN.

Notation

Scalars, vectors, matrices and sets are denoted using lower-case, bold lower-case, bold upper-case and calligraphic upper-case letters, respectively. For a vector $\mathbf{x}$, x_i denotes the i-th entry. For a matrix $\mathbf{A}$, A_{ij} denotes the element of its i-th row and j-th column. The notations $\mathbf{A} \succ \mathbf{0}$ and $\mathbf{A} \succeq \mathbf{0}$ indicate that $\mathbf{A}$ is positive definite and positive semi-definite, respectively. $\mathbf{A}^\top$ denotes the transpose of $\mathbf{A}$ and $\rho(\mathbf{A})$ denotes the spectral radius of matrix $\mathbf{A}$, namely $\rho(\mathbf{A}) = \max\{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_m|\}$, where $\lambda_1, \ldots, \lambda_m$ are the eigenvalues of $\mathbf{A}$. In an iterative process $\mathbf{x}^{(l)} = f(\mathbf{x}^{(l-1)})$, $\mathbf{x}^{(l)}$ denotes the value of $\mathbf{x}$ at iteration (l). $\mathrm{diag}\{\mathbf{x}\}$ denotes the diagonal matrix, whose diagonal entries are the elements of $\mathbf{x}$. $\mathbf{O}$ denotes the matrix whose all entries are zero, the identity matrix is denoted by $\mathbf{I}$ and $\mathbf{1}$ denotes a vector of all ones. $\mathbf{x}_\mathcal{I}$ denotes a subset of the elements of $\mathbf{x}$ according to the set of indices $\mathcal{I}$, namely $\mathbf{x}_\mathcal{I} \subseteq (x_1, x_2, \ldots, x_n)$. For a set $\mathcal{B}$, the notation $\mathcal{B}\backslash i$ denotes all the elements in $\mathcal{B}$ except i. The notation $a \propto b$ denotes that a is proportional to b. $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \mathbf{C})$ denotes that a random variable follows the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{C}$. $\mathcal{R}(\mathbf{A})$ and $\mathcal{N}(\mathbf{A})$ denote the rangespace and nullspace of matrix $\mathbf{A}$, respectively. $\dim \mathcal{V}$ denotes the dimension of vector space $\mathcal{V}$, namely the number of vectors in any of its bases.

Chapter 2

Algorithms

In this Chapter at first we are going to set the basis for *Probabilistic Graphical Models* and explain what *Inference* is. Later, we are going to present the *Gaussian Belief Propagation* algorithm and its application on solving linear systems of equations.

2.1 Inference Algorithms and Probabilistic Graphical Models

2.1.1 Inference Algorithms

Inference is the procedure of learning about the generically hidden state of the world that we care about from available observations.

Consider a collection of random variables $\mathbf{x} = (x_1, \ldots, x_N)$ and the observations about them be presented by random variables $\mathbf{y} = (y_1, \ldots, y_N)$. Let each of these random variables x_i, $1 \leq i \leq N$, take on a value in $\mathcal{X}$ and each observation variable y_i, $1 \leq i \leq N$, take on a value in $\mathcal{Y}$. Given observations $\mathbf{y}$, our goal is to say something about possible realizations of $\mathbf{x}$ [15]. Given this setup, there are two primary computation problems of interest:

1. Calculating *posterior* beliefs,

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} = \frac{p(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{x}' \in \mathcal{X}^N} p(\mathbf{x}', \mathbf{y})}. \tag{2.1}$$

In general computing the denominator of Equation 2.1 is expensive. If we are thinking of N variables then this starts scaling like $|\mathcal{X}^N|$. That is because, without any additional structure, a distribution over N variables, with each variable taking on values in $\mathcal{X}$, requires storing a table of size $|\mathcal{X}|^N$, where each entry contains the probability of a particular realization. Thus, computing *posterior* has complexity exponential in the number of variables N.

2. Calculating the *maximum a posteriori (MAP)* estimate,

$$\begin{aligned}\hat{\mathbf{x}} &\in \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^N} p(\mathbf{x}|\mathbf{y}) \\ &= \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^N} \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})} \\ &= \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}^N} p(\mathbf{x},\mathbf{y})\end{aligned} \tag{2.2}$$

   As before, without any additional structure, the above optimization problem requires searching over the entire space $\mathcal{X}^N$, resulting in an exponential complexity in the number of variables N.

Suppose that the N random variables are now independent, i.e.

$$p(x_1, \ldots, x_n) = p(x_1) \cdot \ldots \cdot p(x_N). \tag{2.3}$$

Then posterior belief calculation can be done separately for each variable. Computing the posterior belief of a particular variable has complexity $|\mathcal{X}|$. Similarly, MAP estimation can be done by finding each variable's assignment that maximizes its own probability. This is done for N variables, so the computational complexity of MAP estimation drops to $N \cdot |\mathcal{X}|$.

Thus, independence or some form of factorization enables efficient computation of both posterior beliefs and MAP estimation. By exploiting factorizations of joint probability distributions and representing these factorizations via graphical models, we can achieve huge computational efficiency gains.

2.1.2 Probabilistic Graphical Models

Probabilistic Graphical Models use a graph-based representation as the basis of compactly encoding a complex distribution over a high-dimensional space [15].

The three main types of graphical models are [16]:

1. *Directed Graphical Models (or Directed Acyclic Graphs (DAGs) or "Bayesian Networks")*
   A directed graphical model is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which consists of nodes $\mathcal{V}$, which represent random variables, and directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The notation $(i, j) \in \mathcal{E}$ means that there is a directed edge from i to j. By choosing a *topological ordering* of the node (i.e. an ordering where any node i comes after all of its parents), then the graph $\mathcal{G}$ implies the conditional independence

$$x_i \perp x_{\nu_i} | x_{\pi_i}, \tag{2.4}$$

   where ν_i is the set of nodes that are not parents of i but they appear in the topological ordering before i. Hence, DAGs define families of distributions which factor by functions of

nodes and their parents. In particular, we assign to each node i a random variable x_i and a non-negative-valued function $f_i(x_i, x_{\pi_i})$ such that,

$$\sum_{x_i \in \mathcal{X}} f_i(x_i, x_{\pi_i}) = 1,$$
$$\prod_i f_i(x_i, x_{\pi_i}) = p(x_1, \ldots, x_N),$$

where π_i denotes the set of parents of node i. The graph is acyclic, thus we must have $f_i(x_i, x_{\pi_i}) = p(x_i | x_{\pi_i})$.

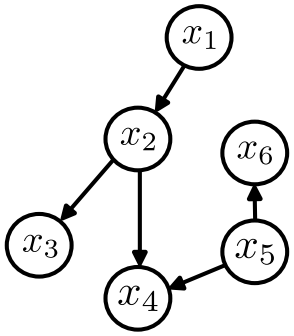


Figure 2.1: An example of a directed acyclic graph representing a distribution with 6 random variables.

2. *Undirected Graphical Models or "Markov Random Fields"*
An undirected graphical model is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which consists of nodes $\mathcal{V}$, which represent random variables, and undirected edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. These graphs define a family of probability distributions which satisfy the following graph separation property,

$$x_A \perp x_B | x_C, \tag{2.5}$$

whenever there is no path from any node in A to any node in B which does not pass through any node in C. Unlike directed graphical models, undirected graphical models do not have a natural factorization into a product of conditional probabilities. Instead, we represent the distribution as a product of functions called *potentials*, times a normalization constant. Given the set of variables $x_1, \ldots, x_N$ and a set $\mathcal{C}$ of maximal cliques, we can define the following representation of the joint distribution,

$$p(\mathbf{x}) \propto \prod_{C \in \mathcal{C}} \psi(x_C) \tag{2.6}$$
$$= \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi(x_C), \tag{2.7}$$

where Z is called the *partition function* and is chosen such that it normalizes the probabilities,

$$Z = \sum_{\mathbf{x}} \prod_{C \in \mathcal{C}} \psi(x_C). \tag{2.8}$$

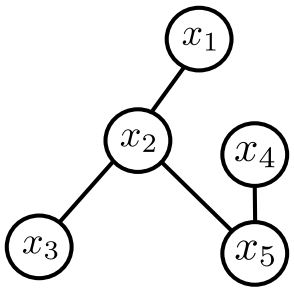


Figure 2.2: An example of a Markov random field representing a distribution with 5 random variables.

3. *Factor Graphs* [17]
   A factor graph consists of a vector of random variables $\mathbf{x} = (x_1, \ldots, x_n)$ and a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, which in addition of variable nodes it consists of factor nodes $\mathcal{F}$. The joint probability distribution associated to a factor graph is given by

$$p(x_1, \ldots, x_n) \propto \prod_{j \in J} g_j(X_j), \tag{2.9}$$

   where J is a discrete index set, X_j is a subset of $\{x_1, \ldots, x_n\}$ and $g_j(X_j)$ is a function having the elements of X_j as arguments.

   The factor graph is a bipartite graph between variable nodes and factor node, i.e. there are no edges connecting a variable node with another variable node or a factor node with another factor node, that expresses the structure of the factorization 2.9. A factor graph has a variable node for each variable x_i, a factor node for each local function g_i and an edge that connects a variable node to a factor node if and only if x_i is an argument of g_i.

   For better understanding of the factor graphs, we provide an example. Let p be a probability density function that can be expressed as the product

$$p(x_1, x_2, x_3, x_4) \propto g_1(x_1, x_3, x_4) \cdot g_2(x_2, x_3, x_4) \tag{2.10}$$

   of two factors, so that $J = \{1, 2\}$, $X_1 = \{x_1, x_2, x_3\}$ and $X_2 = \{x_2, x_3, x_4\}$. The factor graph that corresponds to p is shown in Figure 2.3.

   Factor graphs are very useful in our work, since our main application runs the Gaussian Belief Propagation algorithm on that specific type of graphical models.

2.2 Gaussian Belief Propagation (GBP)

Gaussian Belief Propagation [14] is a special case of the *sum-product* algorithm (or *belief propagation*) where the distributions are Gaussian. The *sum-product* algorithm is a *message passing* algorithm that operates in undirected graphical models or factor graphs. It was firstly introduces by R. G. Gallager [3] during the 1960s and it is still one of the most famous inference algorithms. Given a probability distribution function, sum-product computes -either exactly or approximately-

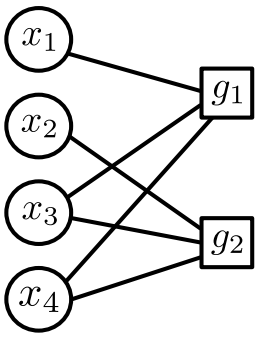


Figure 2.3: A factor graph for the product $p(x_1, x_2, x_3, x_4) \propto g_1(x_1, x_3, x_4) \cdot g_2(x_2, x_3, x_4)$. The variable nodes are represented with circles and the factor node with squares.

various marginal distribution functions using message passing between the graph's nodes and following a few simple computational rules [17, 18].

2.2.1 Gaussian Belief Propagation under High-Order Factorization and Asynchronous Scheduling

Consider a Gaussian random vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^n$. Its probability density function can be written as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}|\Lambda|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Lambda^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{2.11}$$

$$\propto \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Lambda^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{2.12}$$

denoted as $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$, with mean $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ and covariance matrix $\boldsymbol{\Lambda} = \mathbb{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^\top]$. This form of representation is call the *covariance form*. Another very useful form is the *information form*, where the probability density function can be written as

$$p(\mathbf{x}) \propto \exp\left\{-\frac{1}{2}\mathbf{x}^\top \mathbf{J}\mathbf{x} + \mathbf{h}^\top \mathbf{x}\right\}, \tag{2.13}$$

denoted as $\mathbf{x} \sim \mathcal{N}^{-1}(\mathbf{h}, \mathbf{J})$, with potential $\mathbf{h}$ and information (or precision) matrix $\mathbf{J}$. Note that $\mathbf{J} = \boldsymbol{\Lambda}^{-1} \succ \mathbf{0}$ and $\mathbf{h} = \mathbf{J}\boldsymbol{\mu}$.
Conventionally, Gaussian BP is performed under a pairwise factorization of the joint Gaussian pdf

$$p(\mathbf{x}) \propto \prod_i \Phi_i(x_i) \prod_{i \neq j} \Phi_{ij}(x_i, x_j), \tag{2.14}$$

where $\Phi_i(x_i)$ is a function only depending on x_i and $\Phi_{ij}(x_i, x_j)$ is a function depending on x_i and x_j.

Following the work in [19], the joint Gaussian pdf can be written as

$$p(\mathbf{x}) \propto \prod_{i=1}^{n} f_i(x_i) \prod_{j=1}^{m} g_j(\mathcal{X}_j), \tag{2.15}$$

where $f_i(x_i)$ is a function of x_i and $g_j(\mathcal{X}_j)$ is a function of a set of variables $\mathcal{X}_j \subseteq \{x_1, x_2, \ldots, x_n\}$. If at least one function $g_j(\mathcal{X}_j)$ contains more than two variables, it is referred to as a *high-order factorization* of the joint Gaussian pdf. We can consider a high-order factorization with $\mathbf{J} = \mathbf{\Lambda} + \mathbf{\Xi}^\top \mathbf{\Sigma} \mathbf{\Xi}$ and $\mathbf{h} = \mathbf{\Lambda}\boldsymbol{\xi} + \mathbf{\Xi}^\top \mathbf{\Sigma} \mathbf{u}$, where $\mathbf{\Lambda} \triangleq diag(\eta_1, \eta_2, \ldots, \eta_n)$, $\mathbf{\Sigma} \triangleq diag(\zeta_1, \zeta_2, \ldots, \zeta_m)$ with $\eta_i \geq 0$, $\zeta_j > 0$, $\mathbf{\Xi} \in \mathbb{R}^{m \times n}$, $\boldsymbol{\xi} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$. Under this factorization, the joint Gaussian pdf can be rewritten as

$$\begin{aligned} p(\mathbf{x}) &\propto \exp\left\{-\frac{1}{2}\mathbf{x}^\top \left(\mathbf{\Lambda} + \mathbf{\Xi}^\top \mathbf{\Sigma} \mathbf{\Xi}\right) \mathbf{x} + \left(\mathbf{\Lambda}\boldsymbol{\xi} + \mathbf{\Xi}^\top \mathbf{\Sigma} \mathbf{u}\right)^\top \mathbf{x}\right\} \\ &\propto \exp\left\{-\frac{1}{2}\left(\mathbf{x} - \boldsymbol{\xi}\right)^\top \mathbf{\Lambda} \left(\mathbf{x} - \boldsymbol{\xi}\right)\right\} \exp\left\{-\frac{1}{2}\left(\mathbf{\Xi}\mathbf{x} - \mathbf{u}\right)^\top \mathbf{\Sigma} \left(\mathbf{\Xi}\mathbf{x} - \mathbf{u}\right)\right\} \\ &\propto \prod_{i=1}^{n} \exp\left\{-\frac{1}{2}\eta_i \left(x_i - \xi_i\right)^2\right\} \prod_{j=1}^{m} \exp\left\{-\frac{1}{2}\zeta_i \left(\mathbf{\Xi}_{j,:}\mathbf{x} - u_j\right)^2\right\}, \end{aligned} \tag{2.16}$$

where $\mathbf{\Xi}_{j,:}$ denotes the j-th row of $\mathbf{\Xi}$. Based on Equations 2.15 and 2.16, we have

$$f_i(x_i) \propto \exp\left\{-\frac{1}{2}\eta_i (x_i - \xi_i)^2\right\} \tag{2.17}$$

$$g_j(\mathcal{X}_j) \propto \exp\left\{-\frac{1}{2}\zeta_i \left(\sum_{k \in \mathcal{V}_j} \Xi_{jk} x_k - u_j\right)^2\right\} \tag{2.18}$$

where $\mathcal{X}_j \triangleq \{x_k | \Xi_{jk} \neq 0\}$ and $\mathcal{V}_j \triangleq \{k | x_k \in \mathcal{X}_j\}$, which means that $\mathcal{X}_j$ is the set of variables that are connected to factor j and $\mathcal{V}_j$ is the set of indices of those variables. We can also define $\mathcal{G}_i$, which is the set of indices of factors $\{g_j\}_{j=1}^m$ connected directly to variable x_i in the factor graph. Since we examine GBP on factor graphs, where we have only factor-to-variable node connections and variable-to-factor node connections, the expressions of GBP under high-order factorization only require factor-to-variable messages, $m_{g_j \to x_i}^{(l)}(x_i)$, and variable-to-factor messages, $m_{x_i \to g_j}^{(l)}(x_i)$, at each iteration (l). The message update rules of the algorithm under synchronous scheduling are

$$m_{g_j \to x_i}^{(l)}(x_i) \propto \int_{-\infty}^{+\infty} g_j(\mathcal{X}_j) \prod_{k \in \mathcal{V}_j \setminus i} m_{x_k \to g_j}^{(l-1)}(x_k) \, d\mathcal{X}_j \setminus x_i \tag{2.19}$$

$$m_{x_i \to g_j}^{(l)}(x_i) \propto f_i(x_i) \prod_{k \in \mathcal{G}_i \setminus j} m_{g_k \to x_i}^{(l)}(x_i). \tag{2.20}$$

By inserting the expression of $m_{x_i \to g_j}^{(l)}(x_i)$ in 2.20 into 2.19, we get

$$m_{g_j \to x_i}^{(l)}(x_i) \propto \int_{-\infty}^{+\infty} g_j(\mathcal{X}_j) \prod_{k \in \mathcal{V}_j \setminus i} \left(f_k(x_k) \prod_{k' \in \mathcal{G}_k \setminus j} m_{g_{k'} \to x_i}^{(l)}(x_k)\right) d\mathcal{X}_j \setminus x_i. \tag{2.21}$$

Without loss of generality, we assume that the factor-to-variable messages $m_{g_{k'} \to x_i}^{(l-1)}(x_k)$ are of Gaussian form with $m_{g_{k'} \to x_i}^{(l-1)}(x_k) \sim \mathcal{N}\left(x_k; \mu_{g_{k'} \to x_i}^{(l-1)}, \frac{1}{\nu_{g_{k'} \to x_i}^{(l-1)}}\right)$, where $\mu_{g_{k'} \to x_i}^{(l-1)}$ and $\nu_{g_{k'} \to x_i}^{(l-1)}$ their mean and precision, respectively. Thus, substituting the expressions of $f_i(x_i)$ and $g_j(\mathcal{X}_j)$ presented in

Eq. 2.17 and 2.18 and the Gaussian form of $m^{(l-1)}_{g_{k'}\to x_i}(x_k)$ in Eq. 2.21 we get that the analytical expression of factor-to-variable node messages

$$m^{(l)}_{g_j\to x_i}(x_i) \propto \int_{-\infty}^{+\infty} \exp\left\{-\frac{1}{2}\zeta_j\left(\sum_{k\in\mathcal{V}_j}\Xi_{jk}x_k - u_j\right)^2\right\}$$
$$\prod_{k\in\mathcal{V}_j\backslash i}\exp\left\{-\frac{1}{2}\left(\eta_k + \sum_{k'\in G_k\backslash k}\nu^{(l-1)}_{g_{k'}\to x_k}\right)x_k^2 + \left(\eta_k\xi_k + \sum_{k'\in\mathcal{G}_k\backslash j}\nu^{(l-1)}_{g_{k'}\to x_k}\mu^{(l-1)}_{g_{k'}\to x_k}\right)x_k\right\} d\mathcal{X}_j\backslash x_i, \tag{2.22}$$

which are proved to maintain their Gaussian form [19], with $m^{(l)}_{g_j\to x_i}(x_i) \sim \mathcal{N}\left(x_i; \mu^{(l)}_{g_j\to x_i}, 1/\nu^{(l)}_{g_j\to x_i}\right)$. Their mean and precision are given by

$$\mu^{(l)}_{g_j\to x_i} = \begin{cases} \Xi^{-1}_{ji}u_j - \sum_{k\in\mathcal{V}_j\backslash i}\dfrac{\Xi^{-1}_{ji}\Xi_{jk}\left(\eta_k\xi_k + \sum_{k'\in\mathcal{G}_k\backslash j}\nu^{(l-1)}_{g_{k'}\to x_k}\mu^{(l-1)}_{g_{k'}\to x_k}\right)}{\eta_k + \sum_{k'\in\mathcal{G}_k\backslash j}\nu^{(l-1)}_{g_{k'}\to x_k}}, \\ \qquad\qquad \text{if } \eta_k + \sum_{k'\in G_k\backslash k}\nu^{(l-1)}_{g_{k'}\to x_k} > 0, \forall k\in\mathcal{V}_k\backslash i \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{2.23}$$

$$\nu^{(l)}_{g_j\to x_i} = \frac{\Xi^2_{ji}}{\zeta_j^{-1} + \sum_{k\in\mathcal{V}_j\backslash i}\Xi^2_{jk}\left(\eta_k + \sum_{k'\in\mathcal{G}_k\backslash j}\nu^{(l-1)}_{g_{k'}\to x_k}\right)^{-1}}, \tag{2.24}$$

respectively.

From the equations above it can be seen that in order to fully describe each message $m^{(l)}_{g_j\to x_i}(x_i)$ at iteration (l) only two values are required: its *mean* $\mu^{(l)}_{g_j\to x_i}$ and its *precision* $\nu^{(l)}_{g_j\to x_i}$. Also, it can be seen that the calculation of each $\nu^{(l)}_{g_j\to x_i}$ only requires other precision parameters $\nu^{(l)}_{g_k\to x_p}$. On the other hand, in order to compute $\mu^{(l)}_{g_j\to x_i}$, both different precisions $\nu^{(l)}_{g_k\to x_p}$ and means $\mu^{(l)}_{g_k\to x_p}$ are required.

Using the updated message $m^{(l)}_{g_j\to x_i}$ and $f_i(x_i)$, the BP belief, $b^{(l)}(x_i)$, of x_i at iteration (l) can be computed as

$$b^{(l)}(x_i) \propto f_i(x_i)\prod_{k\in\mathcal{G}_i} m^{(l)}_{g_k\to x_i}(x_i). \tag{2.25}$$

By inserting the expressions of $f_i(x_i)$ (Equation 2.17) and $m^{(l)}_{g_k\to x_i}(x_i)$ into Equation 2.25, we obtain

$$b^{(l)}(x_i) \propto \exp\left\{-\frac{1}{2}\left(\eta_i + \sum_{k\in\mathcal{G}_i}v^{(l)}_{g_k\to x_i}\right)x_i^2 + \left(\eta_i\xi_i + \sum_{k\in\mathcal{G}_i}v^{(l)}_{g_k\to x_i}\mu^{(l)}_{g_k\to x_i}\right)x_i\right\}. \tag{2.26}$$

As it is proven again in [19], the BP beliefs are valid Gaussian pdfs, with $b^{(l)}(x_i) \sim \mathcal{N}\left(x_i; \epsilon_i^{(l)}, \sigma_i^{(l)}\right)$, where

$$\epsilon_i^{(l)} = \frac{\eta_i \xi_i + \sum_{k \in \mathcal{G}_i} \nu_{g_k \to x_i}^{(l)} \mu_{g_k \to x_i}^{(l)}}{\eta_i + \sum_{k \in \mathcal{G}_i} \nu_{g_k \to x_i}^{(l)}}, \tag{2.27}$$

$$\sigma_i^{(l)} = \frac{1}{\eta_i + \sum_{k \in \mathcal{G}_i} \nu_{g_k \to x_i}^{(l)}}. \tag{2.28}$$

Assuming that $\nu_{g_j \to x_i}^{(l)}$ are initialized in a way that guarantees convergence we can replace $\nu_{g_{k'} \to x_k}^{(l-1)}$ with $\nu_{g_{k'} \to x_k}^{*}$ in Equation 2.23. Then if $\boldsymbol{\mu}^{(l)}$ stacks all $\mu_{g_j \to x_i}^{(l)}$ at iteration (l), GBP can be reduced to the *synchronous* affine fixed point problem:

$$\boldsymbol{\mu}^{(l)} = \mathbf{A}\boldsymbol{\mu}^{(l-1)} + \mathbf{c}, \tag{2.29}$$

where $\mathbf{A}$ is an $|\mathcal{E}| \times |\mathcal{E}|$ matrix such that $\mathbf{A}\boldsymbol{\mu}^{(l-1)}$ is a column vector containing elements

$$\alpha_{ij} = \begin{cases} -\sum_{k \in \mathcal{V}_j \backslash i} \frac{\Xi_{ji}^{-1} \Xi_{jk} \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*} \mu_{g_{k'} \to x_k}^{(l-1)}}{\eta_k + \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*}} & , \text{ if } \eta_k + \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*} > 0, \forall k \in \mathcal{V}_j \backslash i \\ 0 & , \text{ otherwise} \end{cases} \tag{2.30}$$

and $\mathbf{c}$ is a column vector containing elements

$$\beta_{ij} = \begin{cases} -\sum_{k \in \mathcal{V}_j \backslash i} \frac{\Xi_{ji}^{-1} \Xi_{jk} \eta_k \xi_k}{\eta_k + \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*}} & , \text{ if } \eta_k + \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*} > 0, \forall k \in \mathcal{V}_j \backslash i \\ 0 & , \text{ otherwise} \end{cases} \tag{2.31}$$

As we will prove, the elements of matrix $\mathbf{A}$ can be found using the following,

$$\mathbf{A}(ij, kk') = \begin{cases} -\frac{\Xi_{ji}^{-1} \Xi_{jk} \nu_{g_{k'} \to x_k}^{*}}{\eta_k + \sum_{k'' \in \mathcal{G}_k \backslash j} \nu_{g_{k''} \to x_k}^{*}} & , \text{ if } \eta_k + \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*} > 0, \forall k \in \mathcal{V}_j \backslash i \\ & \text{and } k \in \mathcal{V}_j \backslash i \text{ and } k' \in \mathcal{G}_k \backslash j \\ 0 & , \text{ otherwise.} \end{cases} \tag{2.32}$$

That is because,

$$\begin{aligned} a_{ij} &= \sum_k \sum_{k'} \mathbf{A}(ij, kk') \mu_{g_{k'} \to x_k}^{(l-1)} = \sum_{k \in \mathcal{V}_j \backslash i} \sum_{k' \in \mathcal{G}_k \backslash j} -\frac{\Xi_{ji}^{-1} \Xi_{jk} \nu_{g_{k'} \to x_k}^{*}}{\eta_k + \sum_{k'' \in \mathcal{G}_k \backslash j} \nu_{g_{k''} \to x_k}^{*}} \mu_{g_{k'} \to x_k}^{(l-1)} \\ &= -\sum_{k \in \mathcal{V}_j \backslash i} \sum_{k' \in \mathcal{G}_k \backslash j} \frac{\Xi_{ji}^{-1} \Xi_{jk} \nu_{g_{k'} \to x_k}^{*} \mu_{g_{k'} \to x_k}^{(l-1)}}{\eta_k + \sum_{k'' \in \mathcal{G}_k \backslash j} \nu_{g_{k''} \to x_k}^{*}} = -\sum_{k \in \mathcal{V}_j \backslash i} \frac{\Xi_{ji}^{-1} \Xi_{jk} \sum_{k' \in \mathcal{G}_k \backslash j} \nu_{g_{k'} \to x_k}^{*} \mu_{g_{k'} \to x_k}^{(l-1)}}{\eta_k + \sum_{k'' \in \mathcal{G}_k \backslash j} \nu_{g_{k''} \to x_k}^{*}}. \end{aligned} \tag{2.33}$$

The expressions of GBP under high-order factorization can be updated utilizing *asynchronous* scheduling. The asynchronous updates can become very useful, as we are going to see in Section 2.3, because they can converge even in cases that synchronous ones do not.

2.2.2 Solving Systems of Linear Equations

Our goal is to solve systems of linear equations

$$\mathbf{M}\mathbf{x} = \mathbf{s}, \tag{2.34}$$

where $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{M} \in \mathbb{R}^{m \times n}$ with $m \geq n$, a full rank matrix, and $\mathbf{s} \in \mathbb{R}^m$, for which the least squares solution is [20]

$$\mathbf{x} = \left(\mathbf{M}^\top \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{s}. \tag{2.35}$$

Setting $\mathbf{\Lambda} = \mathbf{0}_{n \times n},\ \mathbf{\Xi} = \mathbf{M},\ \mathbf{\Sigma} = \mathbf{I},\ \xi = \mathbf{0}$ and $\mathbf{u} = \mathbf{s}$, we get

$$\mathbf{J} = \mathbf{\Lambda} + \mathbf{\Xi}^\top \mathbf{\Sigma} \mathbf{\Xi} \Rightarrow \mathbf{J} = \mathbf{M}^\top \mathbf{M}, \tag{2.36}$$

$$\mathbf{h} = \mathbf{\Lambda}\boldsymbol{\xi} + \mathbf{\Xi}^\top \mathbf{\Sigma} \mathbf{u} \Rightarrow \mathbf{h} = \mathbf{M}^\top \mathbf{s}. \tag{2.37}$$

We have also that

$$\mathbf{h} = \mathbf{J}\boldsymbol{\mu} \Rightarrow \mathbf{M}^\top \mathbf{s} = \mathbf{M}^\top \mathbf{M}\boldsymbol{\mu} \Rightarrow \boldsymbol{\mu} = \left(\mathbf{M}^\top \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{s}. \tag{2.38}$$

Hence, Gaussian Belief Propagation can be utilized for the following distribution:

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}, \mathbf{\Lambda}\right) \\ p(\mathbf{x}) &= \mathcal{N}\left(\mathbf{x}; \left(\mathbf{M}^\top \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{s}, \left(\mathbf{M}^\top \mathbf{M}\right)^{-1}\right) \\ &\propto \exp\left\{-\frac{1}{2}\mathbf{x}^\top \mathbf{M}^\top \mathbf{M}\mathbf{x} + \mathbf{s}^\top \mathbf{M}\mathbf{x}\right\}, \end{aligned} \tag{2.39}$$

whose inference of mean value will yield the desired solution of the problem. Thus, the GBP can be seen as an algorithm that solves the set of linear equations $\mathbf{h} = \mathbf{J}\boldsymbol{\mu}$, as well.

The BP message parameters' initialization under high-order convex decomposition are set to $\nu_{g_j \to x_i}^{(0)} = \Xi_{ji}^2 \zeta_j$ for all $(i, j) \in \mathcal{E}$ and $\boldsymbol{\mu}^{(0)} = \mathbf{0}$.

2.2.3 Message Passing Probabilities of GBP in WSNs

As we have seen in GBP, messages from factor node g_j to variable node x_i depend on the messages that were sent to the neighboring variable nodes of the factor node g_j. This can be formulated as

$$m_{g_j \to x_i}^{(l)} = f\left(m_{g_{k'} \to x_k}^{(l-1)},\ \forall k \in \mathcal{V}_j \setminus i, \forall k' \in \mathcal{G}_k \setminus j\right). \tag{2.40}$$

We define c_{g_j} and c_{x_i} as the WSN terminals that g_j and x_i belong to, respectively, and

$$\mathcal{C}_{i,j} = c_{g_j} \cup c_{x_i} \cup \{c_{x_k} \,|\, \forall k \in \mathcal{V}_j \setminus i\} \tag{2.41}$$

the set of WSN terminals that g_j, x_i and variable nodes with indices $k \in \mathcal{V}_j \setminus i$ belong to. Then we have the following equation for the probability of message $m_{g_j \to x_i}$ being updated correctly,

$$p^{\text{update}}_{m_{g_j \to x_i}} = \prod_{c \in \mathcal{C}_{i,j}} \left(1 - p^{\text{out}}_c\right) \cdot p^{\text{trans}}_{c_{g_j}, c} \tag{2.42}$$

and for $p^{\text{trans}}_{i,j} = 1, \ \forall i, j$, we have

$$p^{\text{update}}_{m_{g_j \to x_i}} = \prod_{c \in \mathcal{C}_{i,j}} \left(1 - p^{\text{out}}_c\right). \tag{2.43}$$

Now, we can create the vector $\tilde{\mathbf{p}}$ containing elements $p^{\text{update}}_{m_{g_j \to x_i}}$ for all $(i, j) \in \mathcal{E}$ arranged first on i and then on j, i.e. containing one element for each edge of the graph. If we diagonalize this matrix, we get the expected value of the matrix $\mathbf{\Psi}$, i.e. $\text{diag}(\tilde{\mathbf{p}}) = \mathbf{P} \equiv \mathbb{E}[\mathbf{\Psi}^{(l)}]$, as it will be defined in Section 2.3.

For better understanding we provide the following example. Let's say we have the following factor graph with the variables $\{x_1, x_2, x_3\}$ and the factors $\{g_1, g_2, g_3, g_4\}$

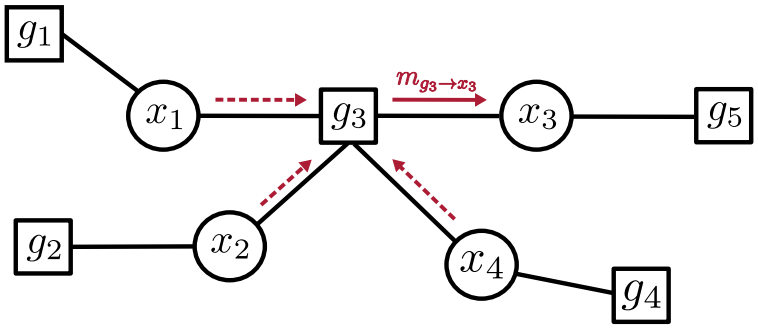


Figure 2.4

There can be many ways to cluster this PGM into 3 WSN terminals. In the next figure we can see one of them.

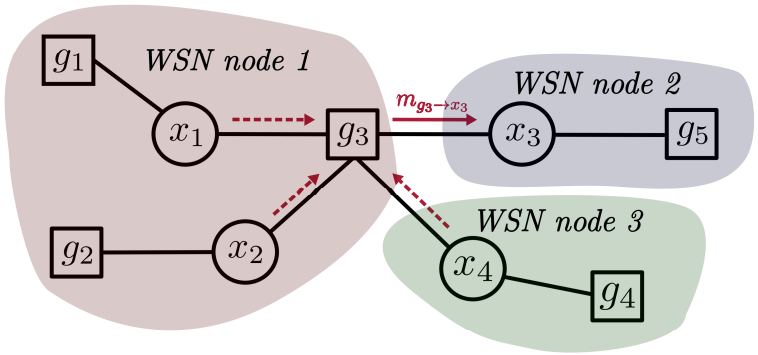


Figure 2.5

We have that $m_{g_3\to x_3}^{(l)} = f\left(m_{g_1\to x_1}^{(l-1)}, m_{g_2\to x_2}^{(l-1)}\right)$, hence probability of update of $m_{g_3\to x_3}^{(l)}$ is,

$$p_{m_{g_3\to x_3}}^{update} = (1-p_1^{\text{out}})(1-p_2^{\text{out}})p_{1,2}^{\text{trans}} = (1-p_1^{\text{out}})(1-p_2^{\text{out}}).$$

2.3 Affine Updates Convergence

In the previous section we saw that GBP can be reduced to the synchronous affine fixed point problem 2.29, $\boldsymbol{\mu}^{(l)} = \mathbf{A}\boldsymbol{\mu}^{(l-1)} + \mathbf{c}$. Due to that result, the need of studying the convergence of the affine fixed point problem arose. We are going to present both *synchronous* and *asynchronous* problems.

2.3.1 Affine Fixed Point (AFP) Problem

Let the real vectors $\mathbf{x}^{(0)},\ \mathbf{b} \in \mathbb{R}^n$ and the real square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. We can from the following recursion

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b}, \quad l = 1, 2, \dots \tag{2.44}$$

The solution of this problem is called the *fixed point* and is denoted by $\mathbf{x}^* \triangleq \lim_{l\to\infty} \mathbf{x}^{(l-1)} = \lim_{l\to\infty} \mathbf{x}^{(l)}$.

Each element of the Equation 2.44 can be expressed as

$$x_k^{(l)} = \sum_{j=1}^{n} a_{kj} x_j^{(l-1)} + b_k, \quad l = 1, 2, \dots \quad k = 1, 2, \dots, n, \tag{2.45}$$

where $x_k^{(l)}$ and b_k denote the k-th element of $\mathbf{x}^{(l)}$ and $\mathbf{b}$, respectively, and a_{ij} the element of the i-th row and j-th column of $\mathbf{A}$. The update of $x_k^{(l)}$ requires all variables $x_j^{(l-1)}$, with respective $a_{kj} \neq 0$, from the previous iteration $(l-1)$ to be known. This constraint results to *synchronous scheduling*, since no output update is possible for any element of the vector $\mathbf{x}$ at a specific iteration, before all necessary input is available for all variables.

This strict constraint can be relaxed. In particular, we can assume that at iteration (l) not all elements of $\mathbf{x}^{(l-1)}$ are readily available for the computation of the corresponding elements of the update vector $\mathbf{x}^{(l)}$. Then, without waiting for all elements to be updated, we keep the corresponding values of the previous iteration and we update the vector $\mathbf{x}^{(l)}$. This results to what we will call *asynchronous scheduling.*

The notation from seminal work in [12, 13, 19] is adopted to formally formulate the above. At each iteration (l), we introduce the functions $\psi_k^{(l)},\ \forall k \in \{1, 2, \dots, n\}$,

$$\psi_k^{(l)} = \begin{cases} 1, \text{ if } x_k \text{ is updated at iteration } (l), \\ 0, \text{ otherwise.} \end{cases} \tag{2.46}$$

Let $\boldsymbol{\psi}^{(l)}$ the binary vector obtained if we stack all $\left\{\psi_k^{(l)}\right\}$, $k \in \{1, 2, \ldots, n\}$, at iteration (l), and $\boldsymbol{\Psi}^{(l)} \triangleq \text{diag}\{\boldsymbol{\psi}^{(l)}\}$ the corresponding diagonal matrix with $\boldsymbol{\psi}^{(l)}$ at its diagonal. This asynchronous formulation alters the Equation 2.44, which becomes

$$\begin{aligned}\mathbf{x}^{(l)} &= \boldsymbol{\Psi}^{(l)}\left(\mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b}\right) + \left(\mathbf{I} - \boldsymbol{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)} \\ &= \left(\boldsymbol{\Psi}^{(l)}\mathbf{A} + \mathbf{I} - \boldsymbol{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)} + \boldsymbol{\Psi}^{(l)}\mathbf{b}.\end{aligned} \tag{2.47}$$

Notice that if $\boldsymbol{\Psi}^{(l)} = \mathbf{I}$, $\forall l$, i.e. $\psi_k^{(l)} = 1$, $\forall l$ and $\forall k$, the asynchronous update of Equation 2.47 reduces to the synchronous one of Equation 2.44.

Based on seminal work in [12, 13], we assume that the aforementioned functions $\psi_k^{(l)}$ are Bernoulli random variables, independent across the different iterations (l) but possibly dependent and not identically distributed across k, with parameters p_k, $\forall k \in \{1, 2, \ldots, n\}$. As a result, we can define the expected value of matrices $\boldsymbol{\Psi}^{(l)}$, $\mathbb{E}[\boldsymbol{\Psi}^{(l)}] \triangleq \mathbf{P} = \text{diag}\{\mathbf{p}\}$, where $\mathbf{p} = \mathbb{E}[\boldsymbol{\psi}^{(l)}]$, a quantity that as we will see plays a major role in the convergence of recursion in Equation 2.47.

2.3.2 Convergence Conditions

2.3.2.1 Convergence of Synchronous AFP Problem

The convergence of the *synchronous* affine fixed point problem of Equation 2.44 has been studied thoroughly in the literature. According to [20, 21], the convergence of the aforementioned update is determined by the spectral radius of matrix $\mathbf{A}$. More specifically, it is shown that a necessary and sufficient condition in order for Equation 2.44 to converge is

$$\rho(\mathbf{A}) < 1. \tag{2.48}$$

2.3.2.2 Convergence of Asynchronous AFP Problem

Much research has also been conducted on the convergence properties of the *asynchronous* affine update of Equation 2.47. In particular, seminal works in [22, 23, 24], which approach the problem from a state-space recursions' perspective, and [19], which studies the convergence of an asynchronous variant of Gaussian Belief Propagation, offer sufficient conditions for mean convergence which are directly applicable to out model.

It can be shown that a sufficient condition for convergence in a mean sense is [19, 22, 23]

$$\rho(\bar{\mathbf{A}}) < 1 \quad \text{and} \quad \rho(\mathbf{S}) < 1, \tag{2.49}$$

where

$$\bar{\mathbf{A}} = \mathbf{P}(\mathbf{A} - \mathbf{I}) + \mathbf{I} \quad \text{and} \quad \mathbf{S} = \bar{\mathbf{A}} \otimes \bar{\mathbf{A}} + ((\mathbf{I} - \mathbf{P})) \otimes \mathbf{P})\mathbf{J}((\mathbf{A} - \mathbf{I}) \otimes (\mathbf{A} - \mathbf{I})), \tag{2.50}$$

with $\mathbf{J} = \sum_{i=1}^{n} \left(\mathbf{e}_i\mathbf{e}_i^{\text{H}}\right) \otimes \left(\mathbf{e}_i\mathbf{e}_i^{\text{H}}\right) = \text{diag}(\text{vec}(\mathbf{I})) \in \mathbb{R}^{n^2 \times n^2}$, $\mathbf{e}_i \in \mathbb{R}^n$ the i-th standard vector that has 1 at the i-th index and 0 elsewhere and $\otimes$ the Kronecker product.

A convergent synchronous AFP problem may diverge with aynchronicity, and conversely, a divergent asynchronous AFP problem may converge simply by the use of asynchronicity. The main difference lies in the notion of convergence. The asynchronous case considers the average behavior and focuses on a mean sense of convergence. The notion of convergence is more relaxed in the asynchronous case and therefore the condition is more relaxed, too [22].

It is also important to note that probabilities of update play an important role in the convergence of asynchronous AFP problem. Namely, the asynchronous system can converge for some set of probabilities, and it may diverge for some set of others. In fact, the rate of convergence can also be increased with the optimal selection of the probabilities similar to other randomized algorithms. These observations will become more clear in Chapter 4.

Chapter 3

Clustering Methods

3.1 k-Means

k-means algorithm was initially introduced by Macqueen in 1967 [25], and is an algorithm that separates a given set of points in a metric space to k distinct clusters. The goal of k-means is to cluster a data set into k groups, so that the sum of distances between each data point and the center of its group is minimized.

In particular, suppose we have the set $\mathbf{S} \in \mathbb{R}^N$ and let $k \in \mathbb{Z}^+$, the number of desired clusters. Define $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\}$ to be a clustering partition of $\mathbf{S}$, i.e. $\mathbf{C}_i \subseteq \mathbf{S}$, with $\mathbf{C}_i \neq \emptyset$, $\forall i = \{1, 2, \ldots, k\}$, $\bigcup_{i=1}^{k} \mathbf{C}_i = \mathbf{S}$ and $\bigcap_{i=1}^{k} \mathbf{C}_i = \emptyset$. Formally, the objective is to solve the optimization problem

$$\begin{aligned} \underset{\mathbf{C}}{\text{argmin}} \quad & f(\mathbf{C}, \boldsymbol{\mu}) = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \mathbf{C}_i} d(\boldsymbol{x}, \boldsymbol{\mu}_i) \\ \text{s.t.} \quad & \bigcup_{i=1}^{k} \mathbf{C}_i = \mathbf{S}, \\ & \bigcap_{i=1}^{k} \mathbf{C}_i = \emptyset, \end{aligned} \tag{3.1}$$

where $\boldsymbol{\mu}_i$ is the average of all points within $\mathbf{C}_i$ and is called *centroid*.
The *Voronoi Region* corresponding to a point $\boldsymbol{\mu}_i \in \mathbb{R}^N$ is the set of all points closest to $\boldsymbol{\mu}_i$, or

$$V(\boldsymbol{\mu}_i) = \{\mathbf{y} \in \mathbb{R}^N \ : \ d(\mathbf{y}, \boldsymbol{\mu}_i) \leq d(\mathbf{y}, \boldsymbol{\mu}_j), \ \forall i \neq i\}$$

In Algorithm 1 we describe the steps of k-means that solves Problem 3.1. The k-means algorithm is based on the following two observations:

- Suppose we have some fixed partition of set $\mathbf{S}$ and we want to pick the best points for minimizing the cost. Clearly, these points are the ones that minimize the distance within each $\mathbf{C}_i$, i.e. the $\boldsymbol{\mu}_i$'s.

- Suppose we have k fixed points $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k$ and we want to find the best partition of $\mathbf{S}$ into k parts. Then this partition well be defining each $\mathbf{C}_i$ to be all of the points in the *Voronoi Region* of $\mathbf{z}_i$, or $\mathbf{C}_i = V(\mathbf{z}_i) \cap \mathbf{S}$.

Algorithm 1: k-Means

Initialize $\boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \ldots, \boldsymbol{\mu}_k^{(0)}$
while $f(\mathbf{C}, \boldsymbol{\mu}) > \epsilon$ **do**
  for $i = 1, 2, \ldots, k$ **do**
    $\mathbf{C}_i = V(\boldsymbol{\mu}_i) \cap \mathbf{S}$
    $\boldsymbol{\mu}_i = \frac{1}{|\mathbf{C}_i|} \sum_{\mathbf{x} \in \mathbf{C}_i} \mathbf{x}$
  end
end

3.2 Spectral Clustering

Spectral clustering has become a very popular clustering algorithm. It is simple to implement, can be solved efficiently by standard linear algebra software, and very often outperforms traditional clustering algorithms such as the k-means algorithm. On the first glance spectral clustering appears slightly mysterious, and it is not obvious to see why it works at all and what it really does. In this section, we will give some intuition on these questions. We will first focus on the math with respect to breaking data into two clusters and later we will generalize it for more clusters. We will only be going through *unnormalized spectral clustering*. For more information on other spectral clustering methods, see [26]. There are also *normalized* spectral clustering algorithms originally proposed by [27] and [28].

3.2.1 Spectral Clustering with 2 Clusters

3.2.1.1 Definitions

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected, weighted graph, with vertex set $\mathcal{V} = \{v_1, \ldots, v_n\}$, with $|\mathcal{V}| = n$. Define the *Weight Matrix*, $\mathbf{W}$, of $\mathcal{G}$, to be an $n \times n$ matrix where w_{ij} represents the weight between vertex v_i and vertex v_j. We make the assumptions about these weights that $w_{ij} \geq 0$, $w_{ii} = 0$ and $w_{ij} = w_{ji}$. Given some $v_i \in \mathcal{V}$, define the *weighted degree* of v_i, as the sum of weights connected directly to v_i,

$$d(v_i) = d_i = \sum_{j=1}^{n} w_{ij}. \tag{3.2}$$

Also, define the *Degree Matrix*, $\mathbf{D}$, of $\mathcal{G}$, as the following diagonal matrix

$$\mathbf{D} = \text{diag}\{d_1, d_2, \ldots, d_n\}. \tag{3.3}$$

Then the *Laplacian Matrix*, $\mathbf{L}$, of $\mathcal{G}$,

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \Rightarrow L_{ij} = \begin{cases} d_i, & \text{if } i = j \\ -w_{ij}, & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0, & \text{otherwise} \end{cases}. \tag{3.4}$$

Suppose that $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ and define $\mathcal{W}(\mathcal{A}, \mathcal{B})$ the sum of the weights connecting A and B,

$$\mathcal{W}(\mathcal{A}, \mathcal{B}) = \sum_{i \in A, j \in B} w_{ij}. \tag{3.5}$$

Definition 3.1. Given $\mathcal{A} \subseteq \mathcal{V}$, define the *cut* of $\mathcal{A}$ to be,

$$cut(\mathcal{A}) = \mathcal{W}(\mathcal{A}, \bar{\mathcal{A}}), \tag{3.6}$$

where $\bar{\mathcal{A}}$ is the complement of $\mathcal{A}$ in $\mathcal{V}$.

Note that $cut(\mathcal{A}) = cut(\bar{\mathcal{A}})$.

We are thinking about how to "best" break a graph into k clusters. One strategy is to minimize the sum of the weights that we cut through between points when separating the graph. Perhaps, we could try finding the minimum $cut(\mathcal{A})$:

$$\underset{\mathcal{A} \subseteq \mathcal{V}}{\text{argmin}} \quad cut(\mathcal{A}). \tag{3.7}$$

It turns out this is easy to solve computationally, but unfortunately, in practice, this tends to separate outliers from the rest of the graph, or simply just separats only one node, that is connected with one edge with the rest of the graph.

We want to try to make $cut(\mathcal{A})$ small but also keep the cardinality of the two components balanced.

Definition 3.2. We can define the *RatioCut* of $\mathcal{A}$ to be:

$$RatioCut(\mathcal{A}) = cut(\mathcal{A}) \cdot \left(\frac{1}{|\mathcal{A}|} + \frac{1}{|\bar{\mathcal{A}}|} \right) = cut(\mathcal{A}) \cdot \left(\frac{1}{|\mathcal{A}|} + \frac{1}{n - |\mathcal{A}|} \right) \tag{3.8}$$

Note that $\left(\frac{1}{|\mathcal{A}|} + \frac{1}{|\bar{\mathcal{A}}|} \right)$ *achieves minimum when* $|\mathcal{A}| \simeq \frac{n}{2}$. *This ratio can be changed if other cluster balance is preferred.*

So, we could try to find the

$$\underset{\mathcal{A} \subseteq \mathcal{V}}{\text{argmin}} \quad RatioCut(\mathcal{A}). \tag{3.9}$$

Now we have got something that sounds more reasonable, but unfortunately for us, this problem is NP-hard. It turns out though that we can work around this issue. With a slight relaxation of this problem, we'll be able to take advantage of the eigenvalues and eigenvectors of the Laplacian to help us. In order to show that, we need to present some more theory.

3.2.1.2 Basic Properties

Given a subset of vertices $\mathcal{A} \subseteq \mathcal{V}$, we define the indicator vector $\mathbb{1}_{\mathcal{A}} = (f_1, \ldots, f_n)^\top \in \mathbb{R}^n$ as the vector with entries $f_i = 1$ if $v_i \in A$ and $f_i = 0$ otherwise.

According to [26], the following hold:

Proposition 3.1. The matrix $\mathbf{L}$ satisfies the following properties:

1. For every vector $\mathbf{f} \in \mathbb{R}^n$ we have

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2} \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2. \tag{3.10}$$

2. $\mathbf{L}$ is positive semi-definite, thus it is also symmetric.
3. The smallest eigenvalue of $\mathbf{L}$ is 0 and the corresponding eigenvector is the constant one vector, $\mathbf{1}$.
4. $\mathbf{L}$ has n non-negative, real-valued eigenvalues, $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Proof. See Appendix B.2. ■

Definition 3.3. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$. The *spectrum* of $\mathbf{A}$, denoted by $\sigma(\mathbf{A})$, corresponds to the set of the *distinct* eigenvalues of $\mathbf{A}$, namely $\sigma(\mathbf{A}) = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$.

Definition 3.4. Suppose $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\lambda \in \sigma(\mathbf{A})$ is an eigenvalue of $\mathbf{A}$.

- The *geometric multiplicity* of λ is equal to $\dim \mathcal{N}(\mathbf{A} - \lambda \mathbf{I})$. In other words, it is the dimension of the span of eigenvectors corresponding to the eigenvalue λ, i.e. the maximal number of linearly independent eigenvectors associated with λ.
- The *algebraic multiplicity* of λ is the number of times the eigenvalue λ is repeated as a root of the characteristic polynomial $\det(s\mathbf{I} - \mathbf{A})$, i.e. the number of times it appears in the spectrum of $\mathbf{A}$.
- If the *algebraic* multiplicity of λ is equal to the *geometric* multiplicity of λ, then λ is called a *semisimple eigenvalue* of $\mathbf{A}$.

Because $\mathbf{L}$ is diagonalizable, it turns out that the *algebraic multiplicity* and the *geometric multiplicity* are equal, so in this section we will just use the term *multiplicity*.

Theorem 3.1 (Number of connected components and the spectrum of $\mathbf{L}$)**.** Let $\mathcal{G}$ be an undirected graph with non-negative weights. Then the *multiplicity* k, of the eigenvalue 0 of $\mathbf{L}$, equals the number of connected components $\mathcal{A}_1, \ldots, \mathcal{A}_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{\mathcal{A}_1}, \ldots, \mathbb{1}_{\mathcal{A}_k}$ of those components.

Proof. See Appendix B.1. ■

Now, we can write the eigenvalues of $\mathbf{L}$ as:

$$0 = \lambda_1 = \ldots = \lambda_k < \lambda_{k+1} \leq \lambda_{k+2} \leq \ldots \leq \lambda_n,$$

where k is the number of connected components of $\mathcal{G}$.

Definition 3.5. Given a graph $\mathcal{G}$ and its Laplacian Matrix $\mathbf{L}$, we define:

- the *Fiedler Value* of $\mathbf{L}$, or *algebraic connectivity* of the graph $\mathcal{G}$, to be the first non-zero eigenvalue
- and *Fiedler Vector* its corresponding eigenvector.

Given a subset $\mathcal{A} \subseteq \mathcal{V}$, we define the vector $\mathbf{f}_{\mathcal{A}} = ((\mathbf{f}_{\mathcal{A}})_1, \ldots, (\mathbf{f}_{\mathcal{A}})_n)^\top \in \mathbb{R}^n$ with entries

$$(\mathbf{f}_{\mathcal{A}})_i = \begin{cases} \sqrt{|\bar{\mathcal{A}}|/|A|} & \text{if } v_i \in \mathcal{A} \\ -\sqrt{|\mathcal{A}|/|\bar{\mathcal{A}}|} & \text{if } v_i \in \bar{\mathcal{A}} \end{cases}$$

Proposition 3.2. The vector $\mathbf{f}_{\mathcal{A}}$ satisfies the following properties:

1. $RatioCut(\mathcal{A}) = \frac{1}{n} \cdot \mathbf{f}_{\mathcal{A}}^\top \mathbf{L} \mathbf{f}_{\mathcal{A}}$,
2. $\mathbf{f}_{\mathcal{A}} \perp \mathbf{1}_n$,
3. $\|\mathbf{f}_{\mathcal{A}}\|^2 = n$.

Proof. See Appendix B.3. ■

3.2.1.3 As an Optimization Problem

Our goal is to solve the optimization problem (3.9),

$$\underset{\mathcal{A} \subseteq \mathcal{V}}{\operatorname{argmin}} \quad RatioCut(\mathcal{A}). \tag{3.11}$$

We can now rephrase the *RatioCut* problem that can be equivalently rewritten as,

$$\begin{aligned} \underset{\mathcal{A}\subseteq\mathcal{V}}{\operatorname{argmin}} \quad & \mathbf{f}_{\mathcal{A}}^{\top}\mathbf{L}\mathbf{f}_{\mathcal{A}} \\ \text{s.t.} \quad & \mathbf{f}_{\mathcal{A}} \perp \mathbf{1}_n \\ & \|\mathbf{f}_{\mathcal{A}}\| = \sqrt{n}. \end{aligned} \tag{3.12}$$

This is a discrete optimization problem as the entries of the solution vector $\mathbf{f}_{\mathcal{A}}$ are only allowed to take two particular values, and of course it is still NP-hard. The most obvious relaxation in this setting is to discard the discreteness condition and instead allow f_i to take arbitrary values in $\mathbb{R}$. This leads to the relaxed optimization problem,

$$\begin{aligned} \underset{\mathbf{f}\in\mathbb{R}^n}{\operatorname{argmin}} \quad & \mathbf{f}^{\top}\mathbf{L}\mathbf{f} \\ \text{s.t.} \quad & \mathbf{f} \perp \mathbf{1}_n \\ & \|\mathbf{f}\| = \sqrt{n}. \end{aligned} \tag{3.13}$$

By the Rayleigh-Ritz theorem [29], it can be seen immediately that the solution of this problem is given by the vector $\mathbf{f}$ which is the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$ (recall that the smallest eigenvalue of $\mathbf{L}$ is 0 with eigenvector 1). So we can approximate a minimizer of *RatioCut* by the Fiedler vector of $\mathbf{L}$. However, in order to obtain a partition of the graph we need to retransform the real-valued solution vector $\mathbf{f}$ of the relaxed problem into a discrete indicator vector. The simplest way to do this is to use the sign of $\mathbf{f}$ as indicator function, that is to choose

$$\begin{cases} v_i \in \mathcal{A}, & \text{if } f_i \geq 0 \\ v_i \in \bar{\mathcal{A}}, & \text{if } f_i < 0 \end{cases}.$$

However, in particular in the case of $k > 2$ treated below, this heuristic is too simple. What most spectral clustering algorithms do instead is to consider the coordinates f_i as points in $\mathbb{R}$ and cluster them into two groups $\mathcal{C}$, $\bar{\mathcal{C}}$ by the k-means clustering algorithm. Then we carry over the resulting clustering to the underlying data points, that is we choose

$$\begin{cases} v_i \in \mathcal{A}, & \text{if } f_i \in \mathcal{C} \\ v_i \in \bar{\mathcal{A}}, & \text{if } f_i \in \bar{\mathcal{C}} \end{cases}.$$

This is the *unnormalized spectral clustering* algorithm for the case of $k = 2$.

3.2.2 Generalized Spectral Clustering

In this section we are going to generalize the Spectral Clustering, for more than 2 clusters.

3.2.2.1 Modified Definitions

The relaxation of the *RatioCut* minimization problem in the case of a general value k, follows a similar principle as the one above, but we first have to extend the definition of *RatioCut* to k dimensions.

Definition 3.6. We can define the *RatioCut* of $\mathcal{A}$, in k dimensions, to be:

$$RatioCut(\mathcal{A}_1, \ldots, \mathcal{A}_k) = \sum_{i=1}^{k} \frac{cut(\mathcal{A}_i)}{|\mathcal{A}_i|}. \tag{3.14}$$

Given a partition $\mathcal{V}$ into k sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$, we define k indicator vectors $\mathbf{h}_j = (h_{1,j}, \ldots, h_{n,j})^\top$ by,

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{|A_j|}}, & v_i \in A_j \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

for $i = 1, 2, \ldots, n$, and define the matrix $\mathbf{H} \in \mathbb{R}^{n \times k}$ to be

$$\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \cdots \ \mathbf{h}_k]. \tag{3.16}$$

Note that $\mathbf{H}$ is an orthogonal matrix by construction, that is $\mathbf{H}^\top \mathbf{H} = \mathbf{I}_n$.

3.2.2.2 Basic Properties

Proposition 3.3. The vectors $\mathbf{h}_j$ and the matrix $\mathbf{H}$ satisfy the following properties:

1. $\mathbf{h}_i^\top \mathbf{L} \mathbf{h}_i = \frac{cut(\mathcal{A}_i)}{|\mathcal{A}_i|}$,
2. $(\mathbf{H}^\top \mathbf{L} \mathbf{H})_{ii} = \mathbf{h}_i^\top \mathbf{L} \mathbf{h}_i$,
3. $RatioCut(\mathcal{A}_1, \ldots, \mathcal{A}_k) = \mathrm{Tr}(\mathbf{H}^\top \mathbf{L} \mathbf{H})$.

Proof. See Appendix B.4. ■

3.2.2.3 As an Optimization Problem

Our goal is to solve the k-dimension optimization problem,

$$\underset{\mathcal{A}_1, \ldots, \mathcal{A}_k}{\text{argmin}} \quad RatioCut(\mathcal{A}_1, \ldots, \mathcal{A}_k). \tag{3.17}$$

We can now rephrase the *RatioCut* problem that can be equivalently rewritten as,

$$\begin{aligned} \underset{\mathcal{A}_1, \ldots, \mathcal{A}_k}{\text{argmin}} \quad & \mathrm{Tr}(\mathbf{H}^\top \mathbf{L} \mathbf{H}) \\ \text{s.t.} \quad & \mathbf{H}^\top \mathbf{H} = \mathbf{I}_n. \end{aligned} \tag{3.18}$$

Similar to above, we now relax the problem by allowing the entries of the matrix $\mathbf{H}$ to take arbitrary real values. Then the relaxed optimization problem becomes,

$$\begin{aligned} \underset{\mathbf{H}\in\mathbb{R}^{n\times k}}{\text{argmin}} \quad & \text{Tr}(\mathbf{H}^\top \mathbf{L}\mathbf{H}) \\ \text{s.t.} \quad & \mathbf{H}^\top\mathbf{H} = \mathbf{I}_n. \end{aligned} \tag{3.19}$$

Again, a version of the Rayleigh-Ritz theorem [29], tells us that the solution is given by choosing $\mathbf{H}$ as the matrix which contains the first k eigenvectors of $\mathbf{L}$ as columns. Again we need to reconvert the real valued solution matrix to a discrete partition. The standard way is to use the k-means algorithm on the rows of $\mathbf{H}$. This leads to the *general unnormalized spectral clustering* algorithm (Algorithm 2).

Algorithm 2: Unnormalized Spectral Clustering

Input: Similarity Matrix, $\mathbf{S} \in \mathbb{R}^{n\times n}$,
Number of Clusters, k

- Construct a similarity graph, as described below. Let $\mathbf{W}$ be its weighted adjacency matrix.
- Compute the unnormalized Laplacian $\mathbf{L}$.
- Compute the first k eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of $\mathbf{L}$.
- Let $\mathbf{U} \in \mathbb{R}^{n\times k}$ be the matrix containing the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ as columns.
- For $i = 1, \ldots, n$, let $\mathbf{y}_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of $\mathbf{U}$.
- Cluster the vectors $(\mathbf{y}_i)_{i=1,\ldots,n} \in \mathbb{R}^k$ with the k-means algorithm into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$.

Output: Clusters $\mathcal{A}_1, \ldots, \mathcal{A}_k$ with $\mathcal{A}_i = \{j \,|\, \mathbf{y}_j \in \mathcal{C}_i\}$.

The similarity matrix is a symmetric and non-negative matrix, with elements the pairwise similarities of our data. When we have a graph that we want to cluster, and not data points that should be used to construct the graph, the similarity matrix is the same with the adjacency matrix of our graph. Then giving as input to Algorithm 2 the adjacency matrix, we can construct the similarity graph by adding edges to connect nodes for which the corresponding value of adjacency matrix is nonzero. For other methods, where we have data points as input, see [26].

3.3 Mapping PGMs to WSN Terminals

A natural question that arises is how to map the PGMs to the different WSN terminals. In this work we consider two different approaches with one of them being more useful in our setting, as detailed below.

The two approaches are:

- *Edge Clustering:* First cluster the *edges* of the PGM and then cluster its *nodes*.
- *Node Clustering:* First cluster the *nodes* of the PGM and then cluster its *edges*.

3.3.1 Edge Clustering

At first, we tried to cluster different parts of vector $\boldsymbol{\mu}$, as in [12], which has one element for each edge of the graph.

3.3.1.1 Previous Method

In [12] the method that is being used is the following. Let

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1^\top \\ \vdots \\ \mathbf{s}_n^\top \end{bmatrix} \in \mathbb{B}^{n \times n} \tag{3.20}$$

denote the binary matrix whose entries are 1 when the corresponding element of matrix $\mathbf{A}$ (as defined in Equations 2.29 and 2.32) is nonzero and 0 otherwise, namely

$$\mathbf{S}_{ij} = \begin{cases} 1, & \mathbf{A}_{ij} \neq 0, \\ 0, & \text{otherwise}, \end{cases} \tag{3.21}$$

where $\mathbf{s}_i^\top$ denotes the i-th row of $\mathbf{S}$. Hence, vectors $\mathbf{s}_i$ are indicative of which elements of $\mathbf{x}$ the update of x_i requires.

Considering all the above, in order to find clusters $\mathbf{C}_i$ and as a result the appropriate assignment to the WSN terminals, they utilize k-means algorithm to the rows of $\mathbf{S}$, $(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n)$, using the ℓ_1-norm as the distance metric $d(\cdot, \cdot)$, namely

$$d(\mathbf{x}, \mathbf{y}) \triangleq \|\mathbf{x} - \mathbf{y}\|_1, \qquad \mathbf{x}, \mathbf{y} \in \mathbb{B}^n. \tag{3.22}$$

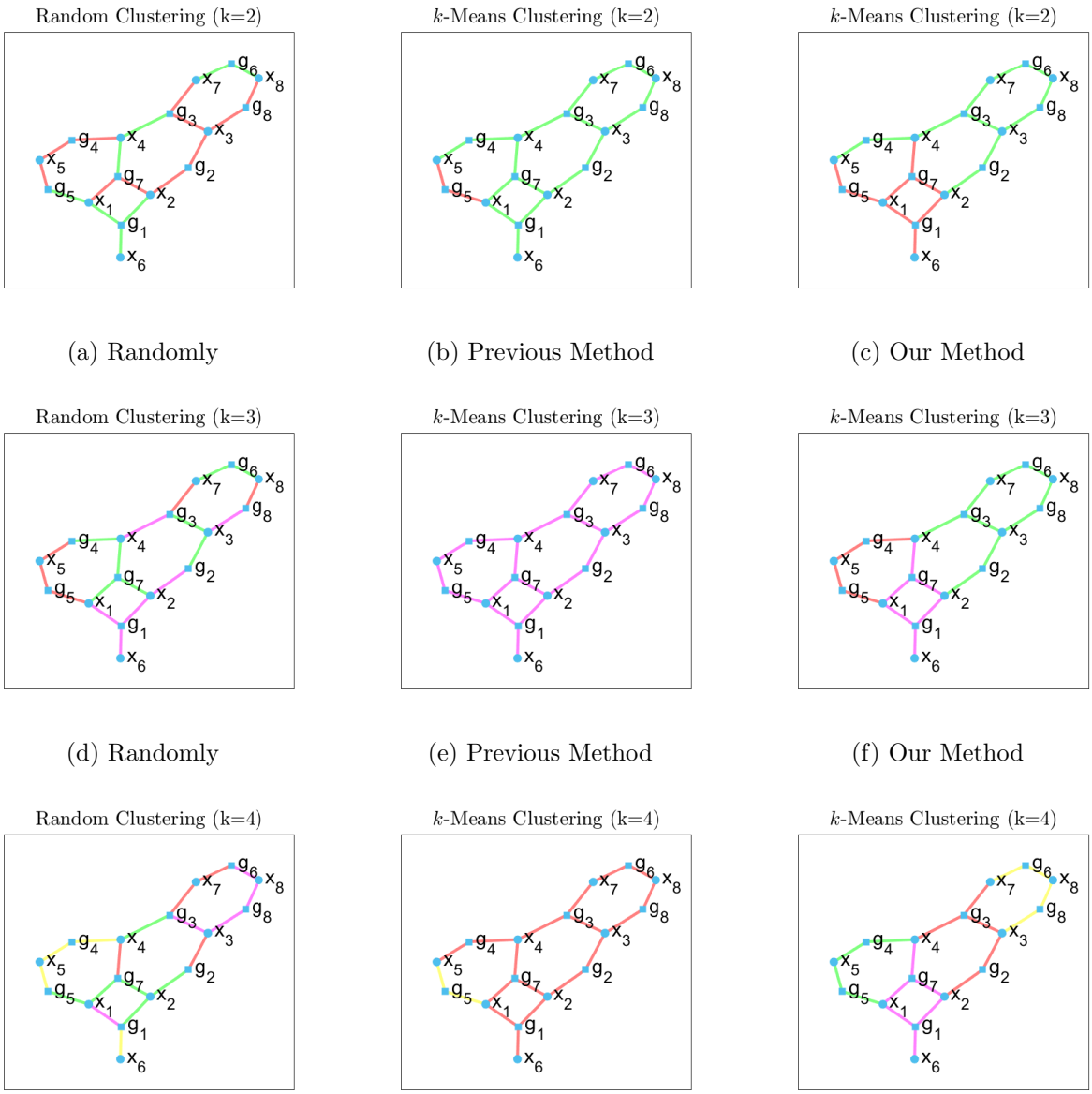
Since the vectors $\mathbf{s}_i$ are binary, then ℓ_1-norm is equivalent to the *Hamming distance*, namely the number of different bits of two vectors.

3.3.1.2 Our Method

Our method, based on this approach, is the following. Let's say that the number of factor nodes in a factor graph, is m. An $m \times m$ matrix $\mathbf{F}$ is created,

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}_1^\top \\ \vdots \\ \mathbf{f}_m^\top \end{bmatrix} \in \mathbb{R}^{m \times m}.$$

Each element $F_{i,j}$ corresponds to the number of common neighboring variable nodes between factor nodes i and j. The reason why this matrix is useful is that when a factor node sends a message to a variable node, it needs the messages that are sent from other factor nodes to its neighboring variable nodes, except the variable node to which it sends the message.

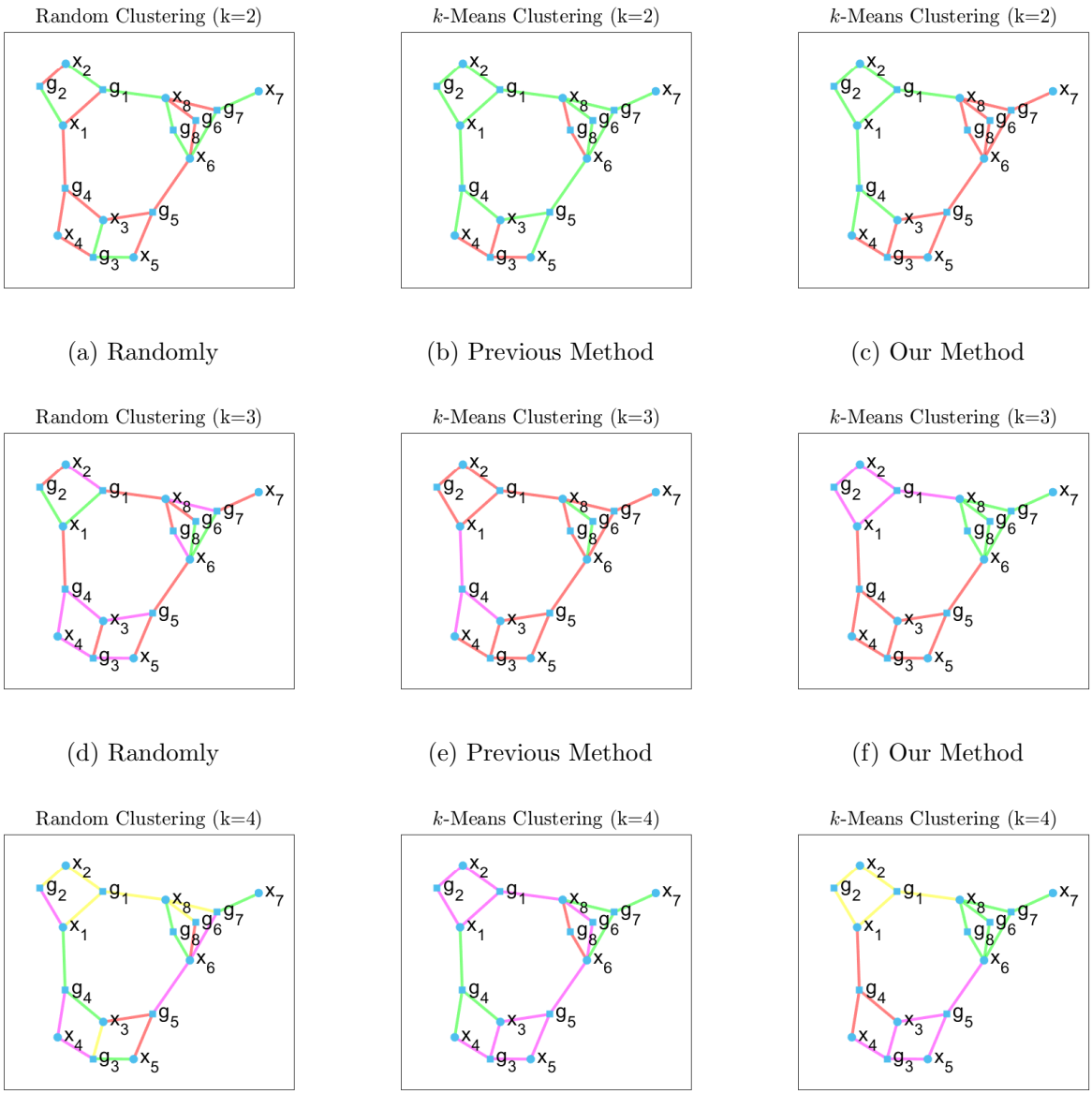


(a) Randomly (b) Previous Method (c) Our Method

(d) Randomly (e) Previous Method (f) Our Method

(g) Randomly (h) Previous Method (i) Our Method

Figure 3.1: Edge Clustering Results for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, for $\mathbf{M} = \mathbf{M}_1$.



(a) Randomly (b) Previous Method (c) Our Method

(d) Randomly (e) Previous Method (f) Our Method

(g) Randomly (h) Previous Method (i) Our Method

Figure 3.2: Edge Clustering Results for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, for $\mathbf{M} = \mathbf{M}_2$.

Considering the above, in order to find clusters $\mathbf{C}_i$ and as a result the appropriate assignment to k WSN terminals, we utilize k-means algorithm, with *squared Euclidean* distance, to the rows of $\mathbf{F}$, $(\mathbf{f}_1, \ldots, \mathbf{f}_m)$. After the clustering of the factor nodes, we cluster the edges of the graph based on which factor node they are connected to.

3.3.1.3 Comparison of Methods

In Figures 3.1 and 3.2 we can see the results of the different clustering methods, compared also with results of a randomly performed clustering, for two different matrices $\mathbf{M}$, as they are defined in Appendix A.

As we can see, k-means as in [12] creates very big clusters with almost all elements of the graph. That is because $\mathbf{A}$ is a matrix that has lots of rows with all their elements equal to 0, which means that they all have 0 distance from each other, thus they are being clustered together. Also, we can see some cases that there are clusters that are empty, which is not preferred for our setting, because we are going to have WSN terminals that are inactive.

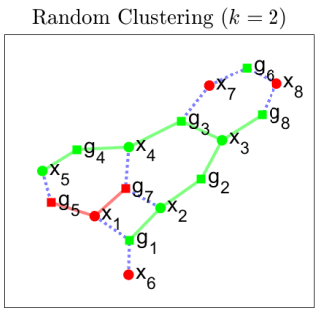
Although our method produced more balanced results than the previous one, there is one more step that needs to be defined to complete the mapping. We have to not only map the edges of the PGM but also its nodes. For nodes that connect edges that belong to the same cluster, this step is easy, we just cluster them in the same cluster with the edges. On the other hand, for nodes that connect edges that belong in different clusters, this procedure becomes more tricky, as we have to decide the cluster of the node, among all the different clusters its connected edges belong to, and furthermore to decide which edges are going to be unclustered so that they can send the required messages between the WSN terminals. Hence, we turned down this approach and we utilized the second one to our problem.

3.3.2 Node Clustering

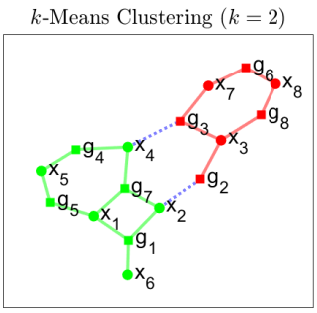
To avoid the problems that we faced using the *Edge Clustering*, we used the *Node Clustering*. Here, we used two different clustering algorithms. The *k-means algorithm* and the *spectral clustering*. The two algorithms are used with different types of inputs.

- To use the *k-means algorithm*, we have to create a data matrix from the given graph, to give it as input to the algorithm. The procedure is as follows. Using the adjacency matrix of the factor graph, we can create the distance matrix $\mathbf{D}$, where $D_{i,j}$ is the length of the *shortest path* between node i and node j. This matrix has size $(n+m) \times (n+m)$ and each row corresponds to a node of the factor graph. Thus, we can cluster the rows of $\mathbf{D}$, $(\mathbf{d}_1, \ldots, \mathbf{d}_{n+m})$, to k clusters, using the k-means algorithm with *squared Euclidean* distance metric.
- In the case of *spectral clustering*, we can directly use the adjacency matrix of the graph, as described in Section 3.2.2.3.

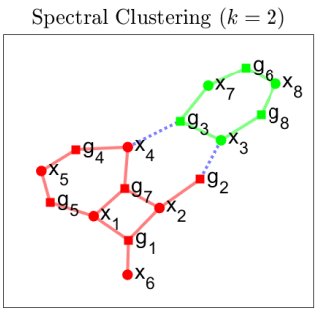
We use the above clustering techniques to cluster the nodes of the graph. Then, the edges which connect two nodes of the same cluster are also clustered to that cluster and the edges that connect



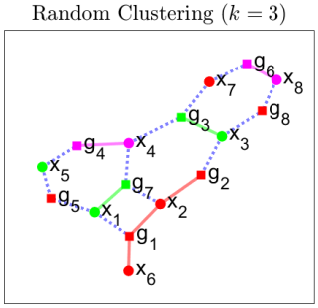
(a) Randomly



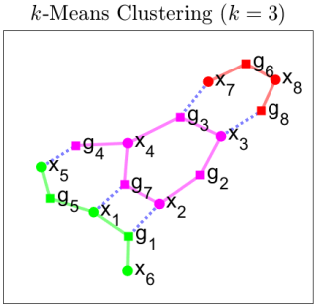
(b) k-Means



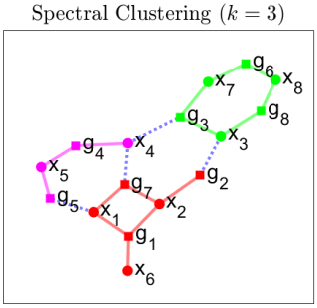
(c) Spectral Clustering



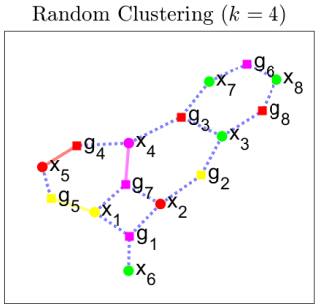
(d) Randomly



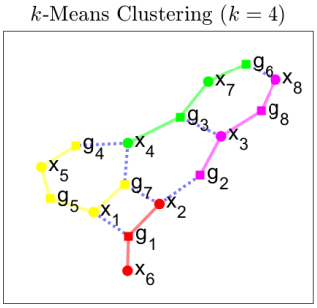
(e) k-Means



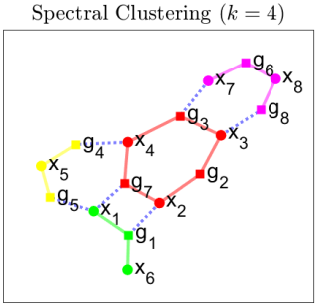
(f) Spectral Clustering



(g) Randomly

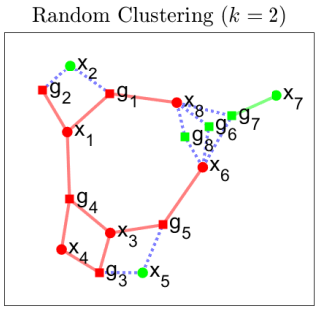


(h) k-Means

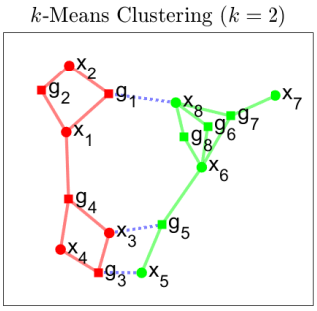


(i) Spectral Clustering

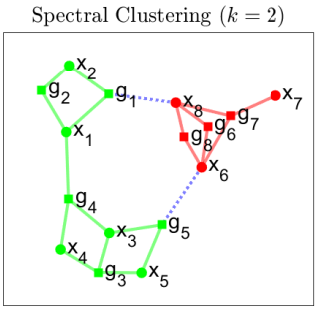
Figure 3.3: Node Clustering Results for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, for $\mathbf{M} = \mathbf{M}_1$.



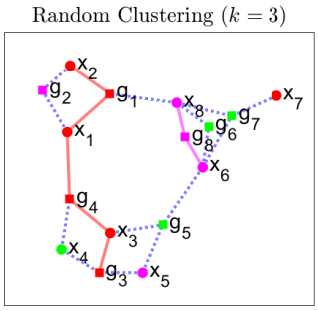
(a) Randomly



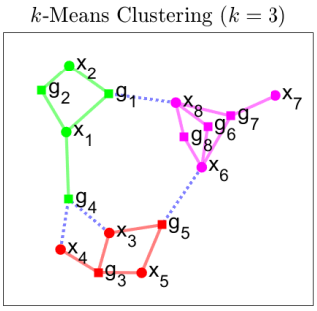
(b) k-Means



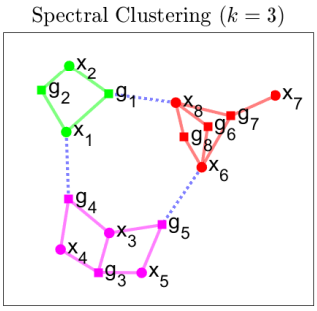
(c) Spectral Clustering



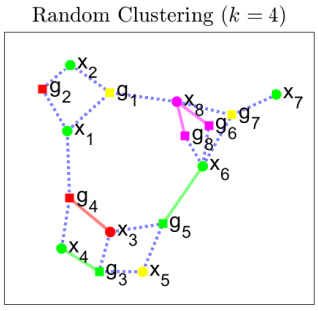
(d) Randomly



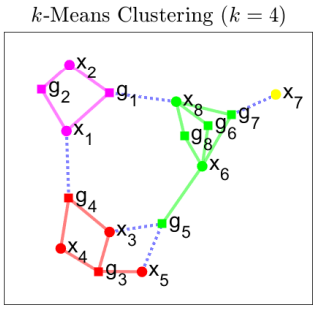
(e) k-Means



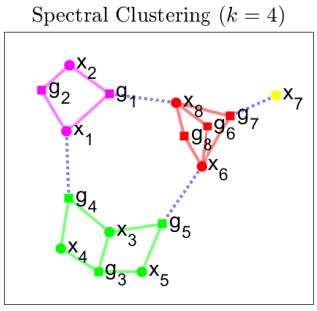
(f) Spectral Clustering



(g) Randomly

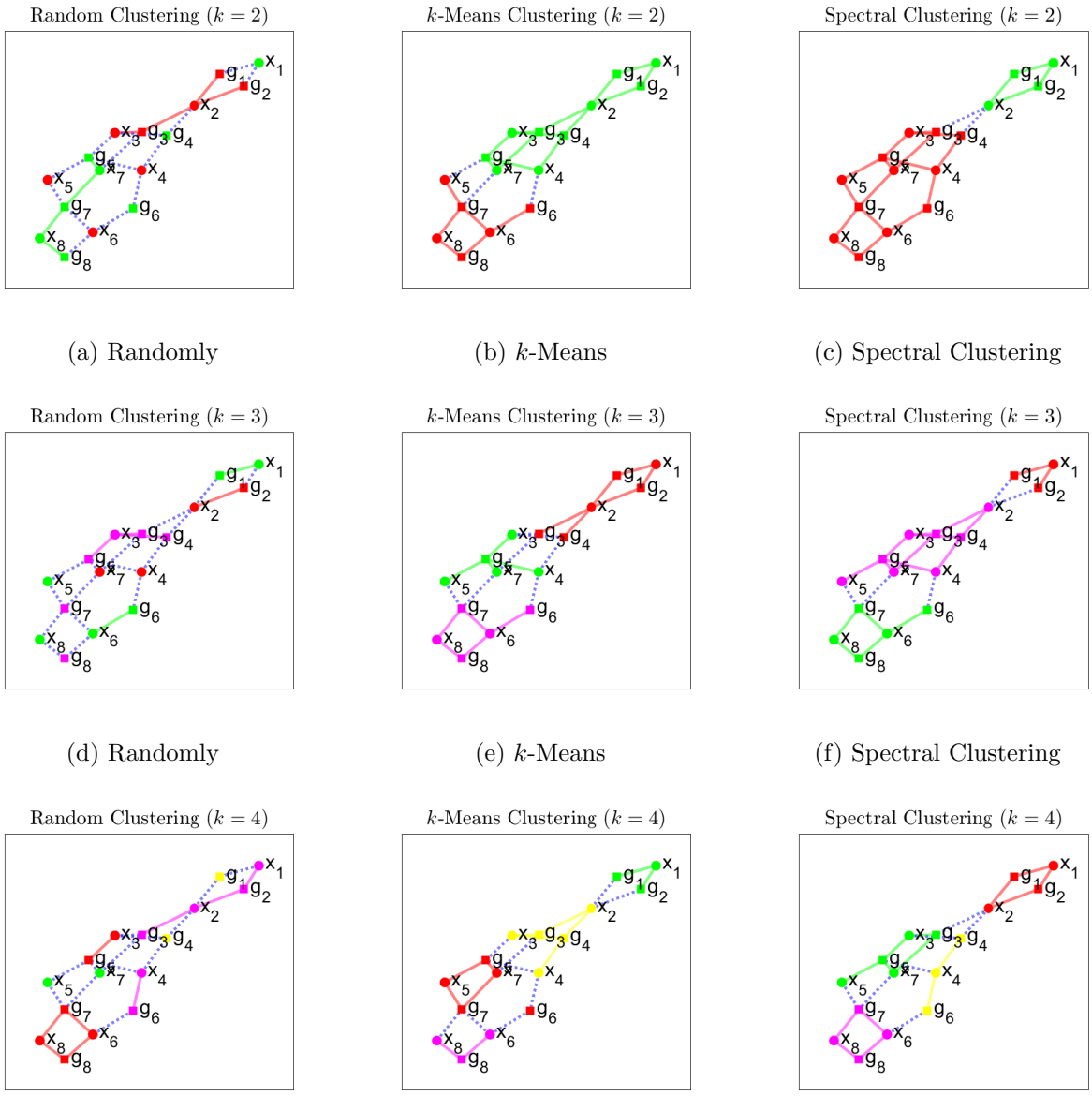


(h) k-Means



(i) Spectral Clustering

Figure 3.4: Node Clustering Results for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, for $\mathbf{M} = \mathbf{M}_2$.

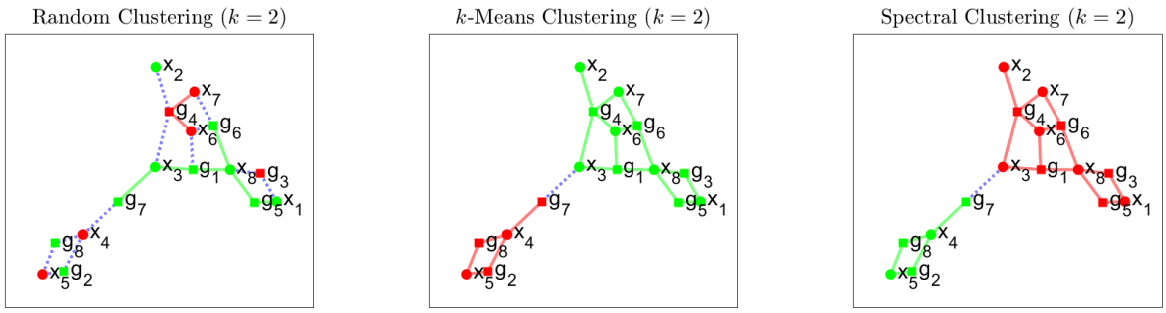


(a) Randomly (b) k-Means (c) Spectral Clustering

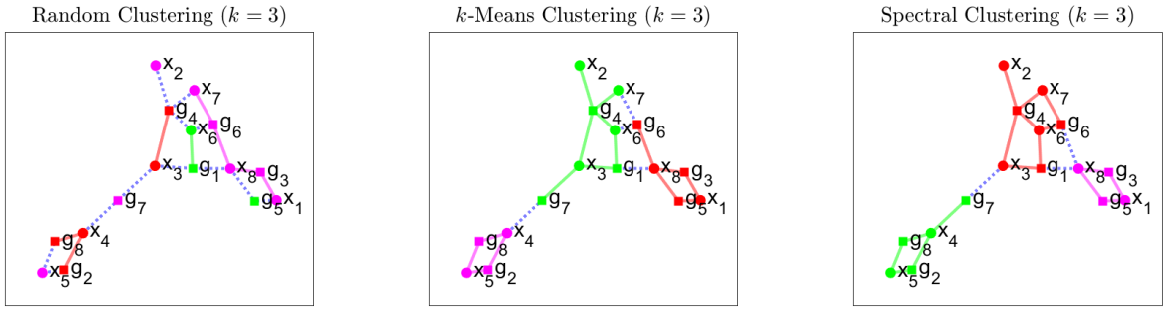
(d) Randomly (e) k-Means (f) Spectral Clustering

(g) Randomly (h) k-Means (i) Spectral Clustering

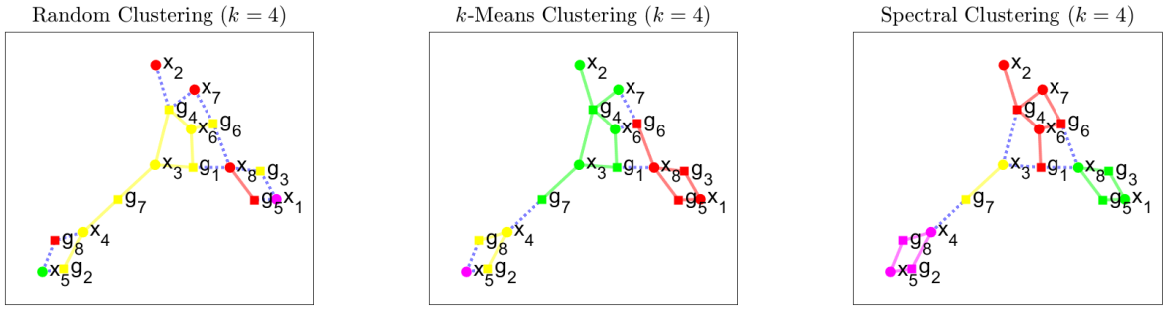
Figure 3.5: Node Clustering Results for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, for $\mathbf{M} = \mathbf{M}_3$.



(a) Randomly (b) k-Means (c) Spectral Clustering



(d) Randomly (e) k-Means (f) Spectral Clustering



(g) Randomly (h) k-Means (i) Spectral Clustering

Figure 3.6: Node Clustering Results for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, for $\mathbf{M} = \mathbf{M}_4$.

nodes of different clusters, stay unclustered. Those unclustered edges are the ones that connect the clusters between them and send the required messages along the WSN terminals.

In Figures 3.3, 3.4, 3.5 and 3.6 we can see the results of clustering the nodes randomly and with k-means and spectral clustering, for various values of k.

To compare the clustering results, we have to define what we consider as "better". This comparison is going to become more clear in Chapter 4, but for now, we are going to consider better the clustering which has less edges connecting different clusters. Hence, we can see that k-means and spectral clustering perform better than random clustering. Also, spectral clustering performs slightly better than k-means.

3.4 Autonomous Clustering

In this section we will consider the problem of clustering in autonomous (ad-hoc) networks and we will present an application of asynchronous graph updates, as proposed in [24]. For this purpose we will combine the well-known spectral clustering [28], with the node clustering method and the polynomial filtering proposed in the following.

3.4.1 Polynomial Filtering

Given a Laplacian matrix $\mathbf{L}$ and an initial vector $\mathbf{x}^{(0)}$, we have the following synchronous affine updates:

$$\mathbf{x}^{(l)} = \mathbf{L}\mathbf{x}^{(l-1)}. \tag{3.23}$$

Vectors $\mathbf{x}^{(l)} \in \mathbb{R}^N$, where N is the number of nodes in the graph, have an element for each node. We will use $nbr(i)$ to denote the neighbors of the i^{th} node. Since $\mathbf{L}$ is a *local* operator, i.e. $L_{i,j} = 0$ when the nodes i and j are not neighbors, a single update can be performed by message passing only between neighboring nodes [24]. That is,

$$x_k^{(l)} = \sum_{j \in nbr(k)} L_{k,j} x_j^{(l-1)}, \quad \forall k. \tag{3.24}$$

Equation 3.23 has a non-zero fixed point if and only if matrix $\mathbf{L}$ has an eigenvalue equal to 1, which is also semisimple and the rest eigenvalues have a magnitude strictly less than 1 [20]. These affine update can converge only to the eigenspace of the unitary eigenvalues of the Laplacian matrix. For the reason that algebraic connectivity λ_2 of a graph is not equal to 1 in nearly every practical example, the affine updates of Equation 3.23 are not directly applicable to our problem.

In order to approach this problem, we will use the r^{th} order polynomial of matrix $\mathbf{L}$,

$$h(\mathbf{L}) = \sum_{n=0}^{r} h_n \mathbf{L}^n, \tag{3.25}$$

for some set of coefficients h_n's. Now, we can consider the affine updates on $h(\mathbf{L})$, as follows:

$$\mathbf{x}^{(l)} = h(\mathbf{L})\mathbf{x}^{(l-1)}. \tag{3.26}$$

The corresponding asynchronous updates are:

$$\mathbf{x}^{(l)} = \mathbf{\Psi}^{(l)}\left(h(\mathbf{L})\mathbf{x}^{(l-1)}\right) + \left(\mathbf{I} - \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)} = \left(\mathbf{\Psi}^{(l)}h(\mathbf{L}) + \mathbf{I} - \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)}, \tag{3.27}$$

with $\mathbf{\Psi}$, as defined in Section 2.3.

Polynomials of a matrix are very useful because of the following two reasons. Firstly, the computation of $(h(\mathbf{L})\mathbf{x})_i$ requires the i^{th} node to retrieve information only from its r-hop neighbors, so if the polynomial is of low order, i.e. r has a small value, then $h(\mathbf{L})\mathbf{x}$ can be computed locally. Secondly, $\mathbf{L}$ and $h(\mathbf{L})$ have the same eigenvectors, that is,

$$h(\mathbf{L}) = \mathbf{V}h(\mathbf{\Lambda})\mathbf{V}^{\mathrm{H}}. \tag{3.28}$$

Therefore, a carefully constructed polynomial can manipulate the eigenvalues of $\mathbf{L}$ in such a way that asynchronous iterations on $h(\mathbf{L})$ can be guaranteed to converge to a desired eigenspace of $\mathbf{L}$ even though iterations on $\mathbf{L}$ itself fail to do so.

Theorem 3.2. [24] Let λ_i denote the eigenvalues of a given matrix $\mathbf{L}$. For a specific target eigenvalue λ_j, assume that a polynomial $h(\cdot)$ satisfies the following conditions:

$$h(\lambda_j) = 1 \quad \text{and} \quad |h(\lambda_i)| < 1 \ \forall \lambda_i \neq \lambda_j. \tag{3.29}$$

Then, asynchronous updates on $h(\mathbf{L})$ as in Equation 3.27 converge to an eigenvector of $\mathbf{L}$ with eigenvalue λ_j for any amount of asynchronicity $0 \leq \delta_T \leq 1$[a].

[a] $\delta_T = \frac{\mathbb{E}[T(N-T)]}{\mathbb{E}[T(N-1)]} = \frac{N-\mu_T-\sigma_T^2/\mu_T}{N-1}$, where μ_T and σ_T^2 denote the mean and the variance of the random quantity T, which is the number of nodes to be updated at each iteration. $\delta_T = 0$ if and only if all the nodes are updated in each iteration, and $\delta_T = 1$ if and only if exactly one node is updated in each iteration.

Theorem 3.2 tells that an *arbitrary* eigenvector of the graph can be computed in a decentralized manner, if a low order polynomial satisfying Equation 3.29 is constructed.

3.4.1.1 The Optimal Polynomial

In this section we consider the construction of the polynomial that has the largest gap between the unit eigenvalue and the rest [24]. In order to represent the condition 3.29 in the matrix-vector form we will use a vector of length $r+1$ to denote the polynomial in Equation 3.25, that is, $\mathbf{h} = [h_0 \cdots h_r]^\top$. Also, let $\mathbf{\Phi}$ be a Vandermonde matrix constructed with the eigenvalues of $\mathbf{L}$ in the following form:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & \lambda_1 & \lambda_1^2 & \cdots & \lambda_1^r \\ 1 & \lambda_2 & \lambda_2^2 & \cdots & \lambda_2^r \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_N & \lambda_N^2 & \cdots & \lambda_N^r \end{bmatrix} \in \mathbb{R}^{N\times(r+1)}. \tag{3.30}$$

In the case of repeated eigenvalues, the repeated rows of the matrix $\mathbf{\Phi}$ are removed. Let ϕ_j denote the row of $\mathbf{\Phi}$ corresponding to the target eigenvalue λ_j, and let $\bar{\mathbf{\Phi}}_j$ denote the remaining rows of $\mathbf{\Phi}$.

In order to find the optimal r^{th} order polynomial satisfying conditions in Theorem 3.2, we consider the following optimization problem:

$$\begin{aligned} \max_{c,\ \mathbf{h}} \quad & c \\ \text{s.t.} \quad & \phi_j \mathbf{h} = 1, \\ & |\bar{\mathbf{\Phi}}_j \mathbf{h}| \leq (1-c)\mathbf{1} \end{aligned} \tag{3.31}$$

where c is a scalar representing the gap between the unit eigenvalue and the rest.

First of all notice that the constraints in the optimization problem 3.31 are linear due to the fact that $\mathbf{\Phi}$ and h are real valued. The objective function is linear as well. Hence, it is a linear programming problem that can be solved efficiently given the eigenvalue matrix $\mathbf{\Phi}$.

The constraints of the problem enforce the polynomial to satisfy the desired condition in Theorem 3.2, while the objective function maximizes the distance between the unit and non-unit eigenvalues of $h(\mathbf{L})$. Therefore, the formulation in the problem searches for the polynomial that yields the fastest rate of convergence among all polynomials of order r satisfying its constraints. Hence, we will refer to the solution of the optimization problem 3.31 as the optimal polynomial.

3.4.1.2 Spectrum-Blind Construction of Suboptimal Polynomials

The locality of the updates needs only to be compromised marginally [24], as the following theorem shows that $r = 2$ is *sufficient* to satisfy the condition in 3.29.

Theorem 3.3. [24] Assume that the matrix $\mathbf{L}$ has real eigenvalues $\lambda_i \in \mathbb{R}$. For a given target eigenvalue λ_j, the condition in 3.29 is satisfied by the following second order polynomial:

$$h(\lambda) = 1 - 2\epsilon(\lambda - \lambda_j)^2 / s_j^2, \tag{3.32}$$

for any ϵ in $0 < \epsilon < 1$ and s_j satisfying the following:

$$s_j \geq \max_{1 \leq i \leq N} |\lambda_i - \lambda_j|. \tag{3.33}$$

Notice that in Equation 3.32 there is a free parameter, ϵ, which can be tuned to increase the gap between the eigenvalues. Thus, the polynomial given in Equation 3.32 is not guaranteed to be optimal in general. It merely shows that a second order polynomial satisfying condition 3.29 always exists, which also implies the feasibility of 3.31 in the case of $r = 2$, or larger.

Although the solution of 3.31 provides the optimal polynomial, it requires the knowledge of all the eigenvalues of $\mathbf{L}$. Such information is not available and difficult to obtain in general. By compromising the optimality, we can construct second order polynomials satisfying 3.29 *without* the knowledge of *all* eigenvalues of $\mathbf{L}$, except the target eigenvalue λ_j.

First of all notice that a value for the coefficient s_j can be found using only the minimum and the maximum eigenvalues of the operator,

$$s_j = \max\{\lambda_{\max} - \lambda_j,\ \lambda_j - \lambda_{\min}\} \tag{3.34}$$

In fact s_j can be satisfied by using an appropriate upper bound for $\lambda_{\max}$ and lower bound for $\lambda_{\min}$. We can use the largest degree $d_{\max}$ of the graph, to select s_j for the following operators.

- *The Laplacian:* the eigenvalues are bounded as $0 \leq \lambda_i \leq 2d_{\max}$. Hence,

$$s_j = d_{\max} + |\lambda_j + d_{\max}|. \tag{3.35}$$

- *The Adjacency:* the eigenvalues are bounded as $-d_{\max} \leq \lambda_i \leq d_{\max}$. Hence,

$$s_j = d_{\max} + |\lambda_j|. \tag{3.36}$$

- *The Normalized Laplacian:* the eigenvalues are bounded as $0 \leq \lambda_i \leq 2$. Hence,

$$s_j = 1 + |\lambda_j - 1|. \tag{3.37}$$

Thus, the polynomial in Equation 3.32 can be constructed using only the target eigenvalue λ_j.

3.4.2 Implementation

As explained in Section 3.2, given a graph, the second smallest eigenvalue of its graph Laplacian, λ_2, is known as the *algebraic connectivity* of the graph [30]. Roughly speaking, graphs with larger λ_2 tend to be more "connected" than others. Furthermore, the corresponding eigenvector $\mathbf{v}_2$, also known as the *Fiedler vector*, can be utilized to cluster the graph into two partitions. The vector $\mathbf{y}$ computed as

$$\mathbf{y} = \text{sign}(\mathbf{v}_2), \tag{3.38}$$

indicates the corresponding clustering of nodes.

In order to compute the eigenvector $\mathbf{v}_2$ of the Laplacian, we utilize the idea of asynchronous polynomial filtering. For this purpose λ_2 will be selected as the target eigenvalue. As a result, nodes will be able to identify the cluster they belong to in an autonomous manner. In this work we make the standing assumption that the *algebraic connectivity* λ_2 is known. Such an assumption is plausible for static networks since it can be inferred in a distributed way by several algorithms [31, 32, 33, 34].

The implementation that we use is not the same as the communication protocol that is described in Algorithm 1 in [24]. Our implementation is based on the asynchronous scheduling that is described in Section 3 and in [12, 13, 19], in the contrary that here $\mathbf{x}$ represents node and not edge values.

At first, we randomly cluster the nodes of the graph to the WSN terminals. Then, we assign probabilities of update for each node based on the terminal they belong to and its probability of being in energy outage. Finally, we perform affine updates utilizing *synchronous* and *asynchronous* scheduling, as in Equations 3.26 and 3.27, respectively.

3.4.3 Results

For the matrices $\mathbf{M}$ in Appendix A, we run the autonomous clustering for 4 different schedulings.

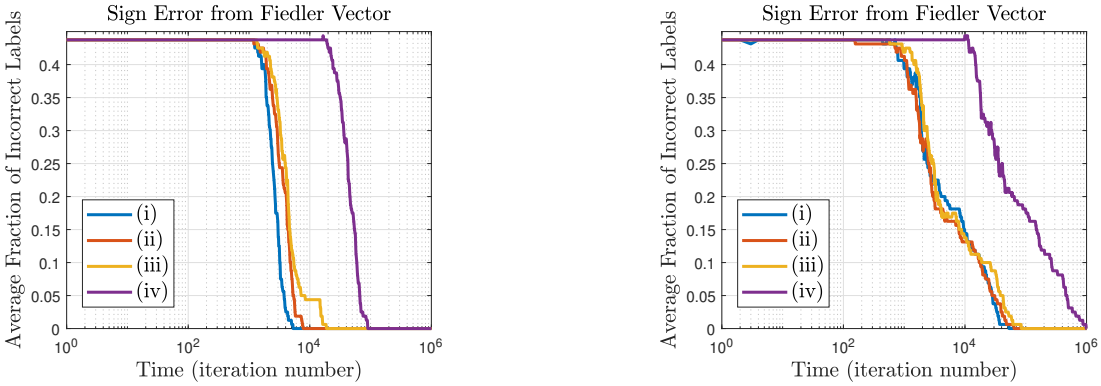
(i) For $p_1 = p_2 = 1$, i.e. *synchronous* scheduling.

(ii) For $p_1 = 0.4958$, $p_2 = 0.8295$, where p_1 and p_2 are the probabilities of terminals 1 and 2, respectively, not being in energy outage.

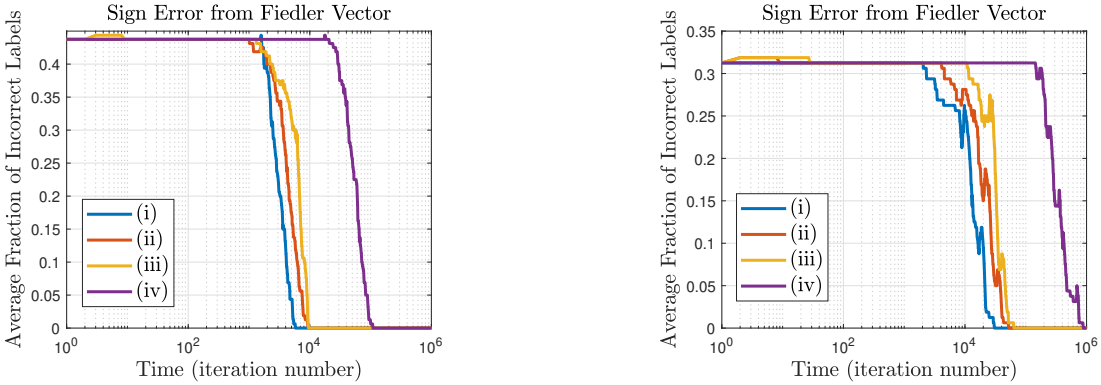
(iii) For $p_1 = p_2 = 0.5$.

(iv) For $\delta_T = 1$, i.e. only one element of $\mathbf{x}$ is being updated in each iteration.

Experimental results of convergence rate of the algorithm can be seen in Figure 3.7 and the clustered graphs in Figure 3.8

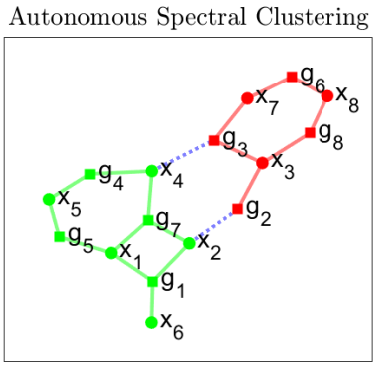


(a) $\mathbf{M} = \mathbf{M}_1$ (b) $\mathbf{M} = \mathbf{M}_2$

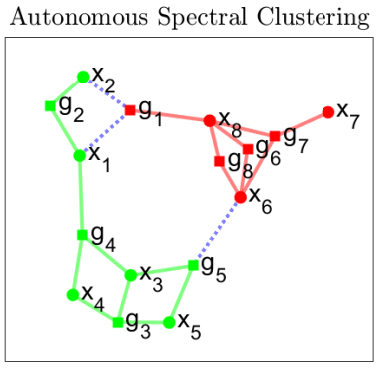


(c) $\mathbf{M} = \mathbf{M}_3$ (d) $\mathbf{M} = \mathbf{M}_4$

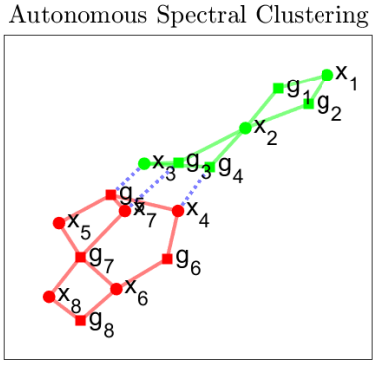
Figure 3.7: Convergence rate of Autonomous Clustering for the graph that is created when GBP is used to solve a linear system of equations $\mathbf{Mx} = \mathbf{s}$, which are obtained by averaging over 10 independent experiments. In the figures we can see the average fraction of incorrect signs of the computed vector from the Fiedler vector.



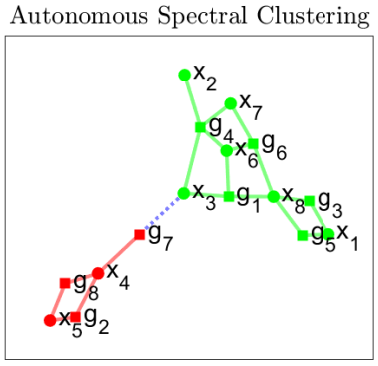
(a) $\mathbf{M} = \mathbf{M}_1$



(b) $\mathbf{M} = \mathbf{M}_2$



(c) $\mathbf{M} = \mathbf{M}_3$



(d) $\mathbf{M} = \mathbf{M}_4$

Figure 3.8: Clustering Results of Autonomous Clustering.

Chapter 4

Simulations

Following Section 2.2.2, our task is to solve the linear system of equations

$$\mathbf{M}\mathbf{x} = \mathbf{s}. \tag{4.1}$$

In all the experiments we assume that $\mathbf{s} = \mathbf{1}$ (all one vector) and $\mathbf{M}$ is one of the matrices that are provided in Appendix A. As described in the aforementioned Section, it is possible to find the solution of 4.1,

$$\mathbf{x} = \left(\mathbf{M}^\top \mathbf{M}\right)^{-1} \mathbf{M}^\top \mathbf{s}, \tag{4.2}$$

inferring the expected value of the distribution

$$p(\mathbf{x}) \propto \exp\left\{-\frac{1}{2}\mathbf{x}^\top \mathbf{M}^\top \mathbf{M}\mathbf{x} + \mathbf{s}^\top \mathbf{M}\mathbf{x}\right\}, \tag{4.3}$$

using Gaussian Belief Propagation.

4.1 Minimization of Convergence Time

4.1.1 Minimization using Clustering

In this section, we present experimental results of the convergence rate of Gaussian Belief Propagation when it is utilized to solve linear systems of equations. We want to compare its performance when the produced graph that GBP uses, is clustered using random clustering, k-means, spectral clustering and autonomous clustering.

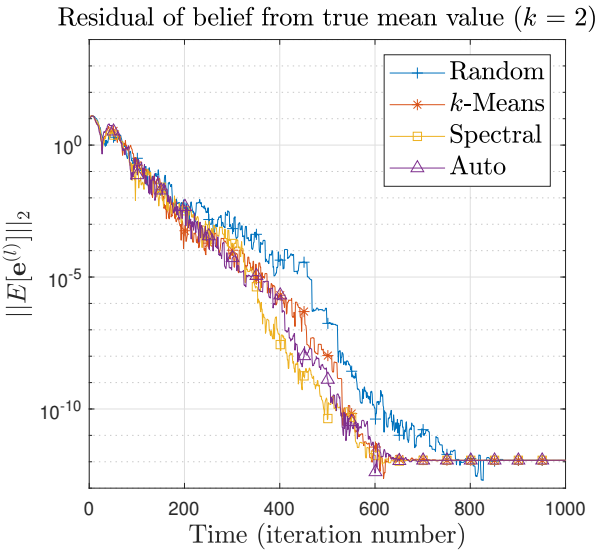
We present experimental results for GBP solving a linear system of equations in an asynchronous manner. In particular, we plot the estimation of the per iteration expected value $\|\mathbb{E}[\mathbf{e}^{(l)}]\|_2$ using 20 independent experiments. At iteration (l), the error $\mathbf{e}$ is defined as

$$\mathbf{e}^{(l)} \triangleq \boldsymbol{\epsilon}^{(l)} - \hat{\mathbf{x}}, \tag{4.4}$$

where $\boldsymbol{\epsilon}^{(l)}$ is the vector that is constructed if we stack all *belief means*, according to Equation 2.27 and $\hat{\mathbf{x}} \triangleq (\mathbf{M}^\top \mathbf{M})^{-1}\mathbf{M}^\top \mathbf{s}$, the least squares solution of the system.

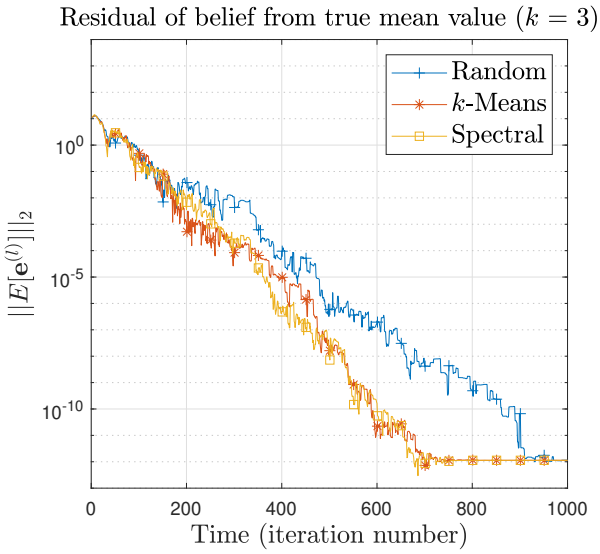
More specifically, Figures 4.1, 4.2, 4.3 and 4.4 present results for random clustering, k-means, spectral clustering and autonomous clustering when we have 2, 3 or 4 WSN terminals. In these figures all the terminals have equal probabilities of update ($p = 0.5$), in order to have comparable results.

We see that clustering the nodes of the PGMs with different clustering methods can *heavily* alter the behavior of the algorithm. For instance, notice in Figures 4.1, 4.2, 4.3 and 4.4 that k-means, spectral and autonomous clustering experiments converge much faster than random clustering. Especially, in most cases, spectral clustering creates a mapping with which GBP has a much faster convergence. This is something that needs further theoretical examination.



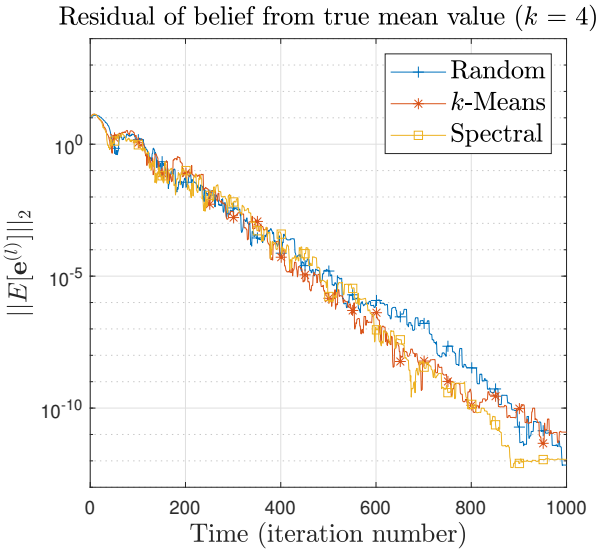
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.922645,\ \rho(\mathbf{S}) = 0.954795$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.916226,\ \rho(\mathbf{S}) = 0.933944$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.916149,\ \rho(\mathbf{S}) = 0.933944$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.916317,\ \rho(\mathbf{S}) = 0.933944$



(b)

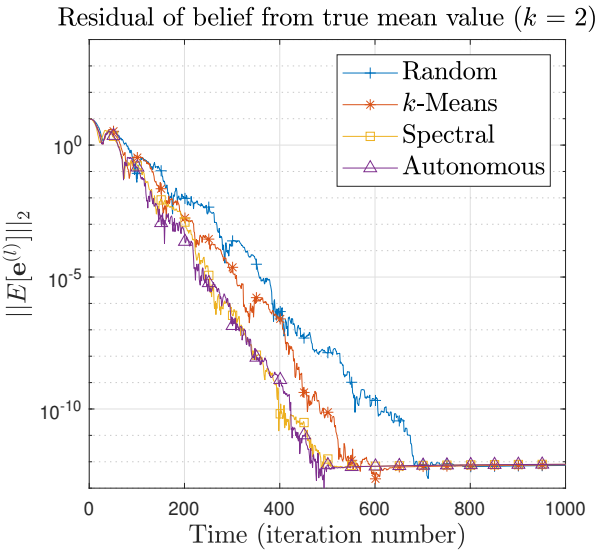
Random	$\rho(\bar{\mathbf{A}}) = 0.954052,\ \rho(\mathbf{S}) = 0.979500$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.933352,\ \rho(\mathbf{S}) = 0.949142$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.930477,\ \rho(\mathbf{S}) = 0.941901$



(c)

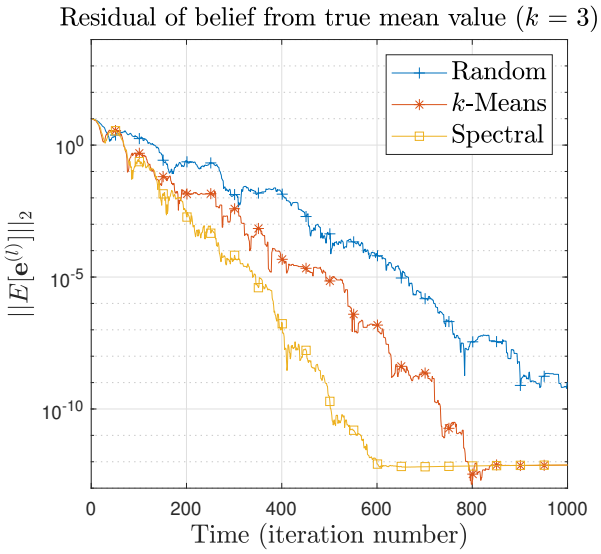
Random	$\rho(\bar{\mathbf{A}}) = 0.961769,\ \rho(\mathbf{S}) = 0.982364$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.953974,\ \rho(\mathbf{S}) = 0.972436$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.948451,\ \rho(\mathbf{S}) = 0.966137$

Figure 4.1: $\mathbf{M}_1$
Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to 0.5.



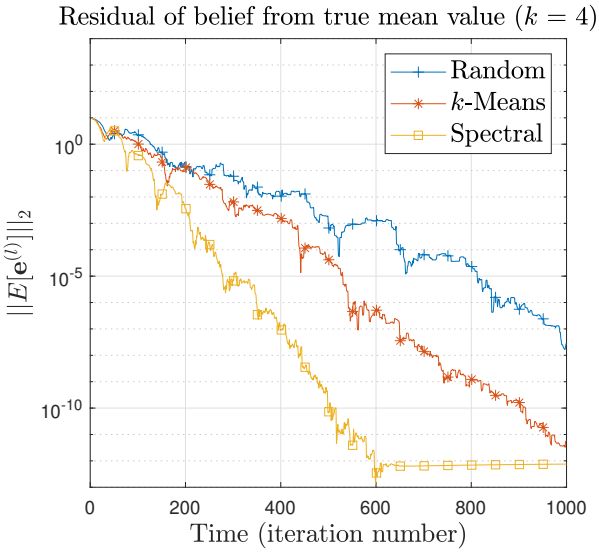
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.934081,\ \rho(\mathbf{S}) = 0.914558$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.927676,\ \rho(\mathbf{S}) = 0.900167$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.921910,\ \rho(\mathbf{S}) = 0.888018$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.921893,\ \rho(\mathbf{S}) = 0.888011$



(b)

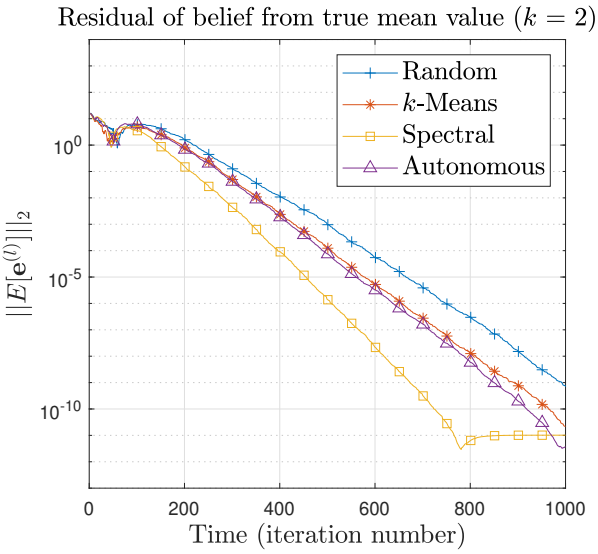
Random	$\rho(\bar{\mathbf{A}}) = 0.951490,\ \rho(\mathbf{S}) = 0.936170$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.933742,\ \rho(\mathbf{S}) = 0.905547$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.933177,\ \rho(\mathbf{S}) = 0.903274$



(c)

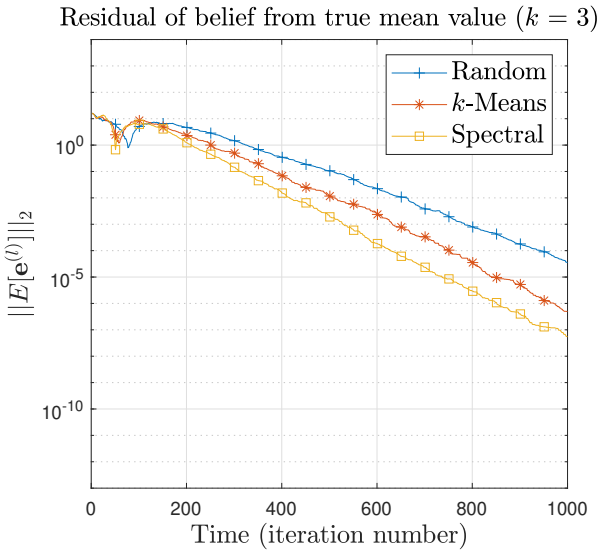
Random	$\rho(\bar{\mathbf{A}}) = 0.950980,\ \rho(\mathbf{S}) = 0.937111$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.947370,\ \rho(\mathbf{S}) = 0.925713$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.933096,\ \rho(\mathbf{S}) = 0.903181$

Figure 4.2: $\mathbf{M}_2$
Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to 0.5.



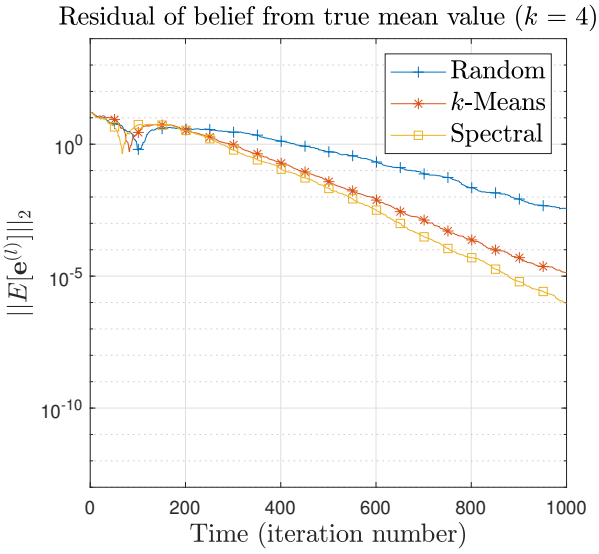
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.970077,\ \rho(\mathbf{S}) = 0.943039$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.967019,\ \rho(\mathbf{S}) = 0.938158$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.958183,\ \rho(\mathbf{S}) = 0.920543$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.966258,\ \rho(\mathbf{S}) = 0.936962$



(b)

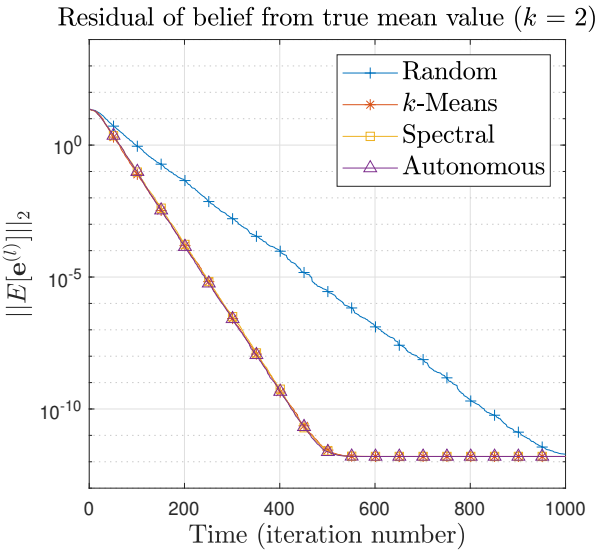
Random	$\rho(\bar{\mathbf{A}}) = 0.980217,\ \rho(\mathbf{S}) = 0.962883$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.972761,\ \rho(\mathbf{S}) = 0.948190$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.969092,\ \rho(\mathbf{S}) = 0.941263$



(c)

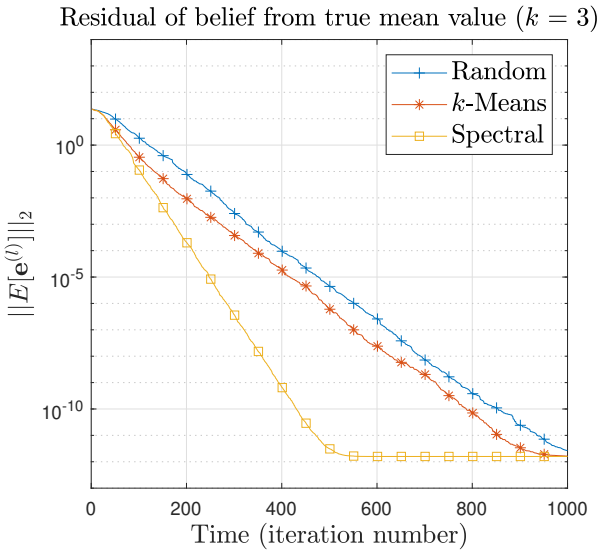
Random	$\rho(\bar{\mathbf{A}}) = 0.984537,\ \rho(\mathbf{S}) = 0.971168$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.977670,\ \rho(\mathbf{S}) = 0.957816$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.976658,\ \rho(\mathbf{S}) = 0.955791$

Figure 4.3: $\mathbf{M}_3$
Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to 0.5.



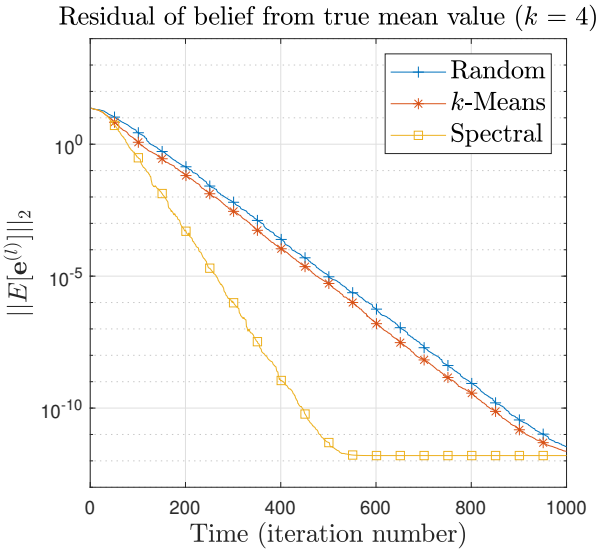
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.954283,\ \rho(\mathbf{S}) = 0.914139$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.939044,\ \rho(\mathbf{S}) = 0.885519$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.939044,\ \rho(\mathbf{S}) = 0.885519$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.939044,\ \rho(\mathbf{S}) = 0.885519$



(b)

Random	$\rho(\bar{\mathbf{A}}) = 0.955807,\ \rho(\mathbf{S}) = 0.917001$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.945140,\ \rho(\mathbf{S}) = 0.896967$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.939044,\ \rho(\mathbf{S}) = 0.885519$



(c)

Random	$\rho(\bar{\mathbf{A}}) = 0.963426,\ \rho(\mathbf{S}) = 0.931312$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.957331,\ \rho(\mathbf{S}) = 0.919864$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.939044,\ \rho(\mathbf{S}) = 0.885519$

Figure 4.4: $\mathbf{M}_4$
Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to 0.5.

4.1.2 Minimization using Spectral Radius

Intuitively, there should be a link between the spectral radius of the iteration matrices $\mathbf{A}$ and $\bar{\mathbf{A}}$ (as defined in Section 2.3), for the synchronous and asynchronous cases, respectively, and the rate of convergence. In the following, we present proofs for a special case of matrices, those with all their eigenvectors independent (which is something that holds for most of the matrices). Generally, for various matrices, we observe the same thing experimentally, which is something that holds for the matrix $\mathbf{S}$, too. Matrices $\mathbf{A}$, $\bar{\mathbf{A}}$ and $\mathbf{S}$ are defined in Section 2.3.

4.1.2.1 Theory

Theorem 4.1. *(Synchronous Case)*
Suppose that the $n \times n$ matrix $\mathbf{A}$ has n linearly independent eigenvectors, then the Synchronous Affine Updates converge faster to the fixed point, for smaller spectral radius, $\rho(\mathbf{A})$.

Proof. See Appendix B.5. ■

Theorem 4.2. *(Asynchronous Case)*
Suppose that the $n \times n$ matrix $\mathbf{A}$ has n linearly independent eigenvectors, then the Asynchronous Affine Updates converge faster to the fixed point, for smaller spectral radius, $\rho(\bar{\mathbf{A}})$.

Proof. See Appendix B.6. ■

In Theorem 4.2 the convergence of the asynchronous case considers the average behavior of the affine updates, as the norm of the *expected value* of the error is the quantity that is being observed. The concept of the different notion in convergence, between synchronous and asynchronous scheduling, is further described in Section 2.3.

4.1.2.2 Numerical Results

The above theoretical results can be shown experimentally, too. For synchronous scheduling, we can see in Figure 4.5, that the affine updates,

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b}, \tag{4.5}$$

converge faster to the fixed point, for smaller spectral radius, $\rho(\mathbf{A})$, confirming Theorem 4.1. As in the synchronous case, in Figure 4.6 we can also see that in the asynchronous case the affine updates,

$$\mathbf{x}^{(l)} = \left(\mathbf{\Psi}^{(l)}\mathbf{A} + \mathbf{I} - \mathbf{\Psi}^{(l)}\right)\mathbf{x}^{(l-1)} + \mathbf{\Psi}^{(l)}\mathbf{b}, \tag{4.6}$$

where $\mathbb{E}\left[\mathbf{\Psi}^{(l)}\right] \equiv \mathbf{P}$, $\forall l$, converge faster to the fixed point, for smaller spectral radius, $\rho(\bar{\mathbf{A}})$, confirming Theorem 4.2. Furthermore, we can see that the same holds for $\rho(\mathbf{S})$, something that we have not proven theoretically yet.

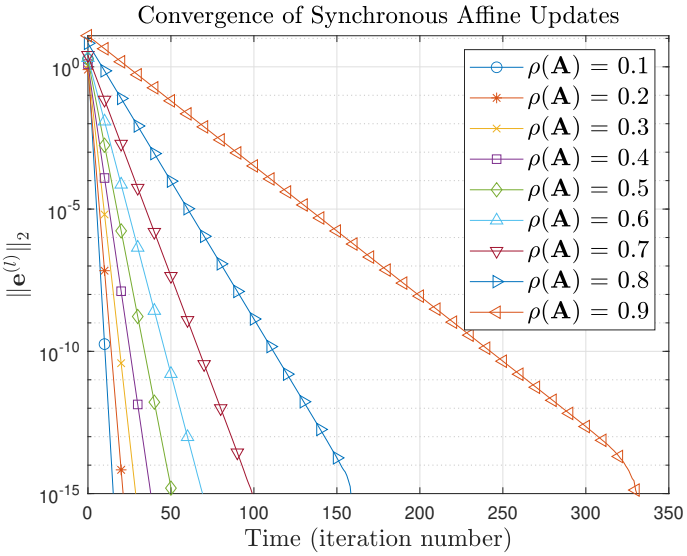
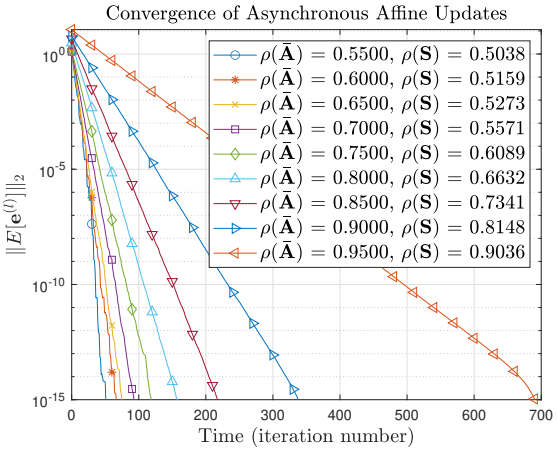
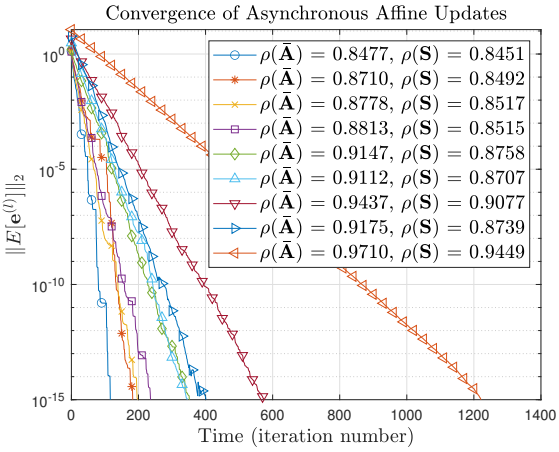


Figure 4.5: Convergence rate of the synchronous affine fixed point problem for different values of $\rho(\mathbf{A})$, for random matrices $\mathbf{A}$.



(a) $\mathbf{P} = 0.5 \cdot \mathbf{I}$



(b) For $\mathbf{P} = \text{diag}([0.6620, 0.1549, 0.8736, 0.2920])$

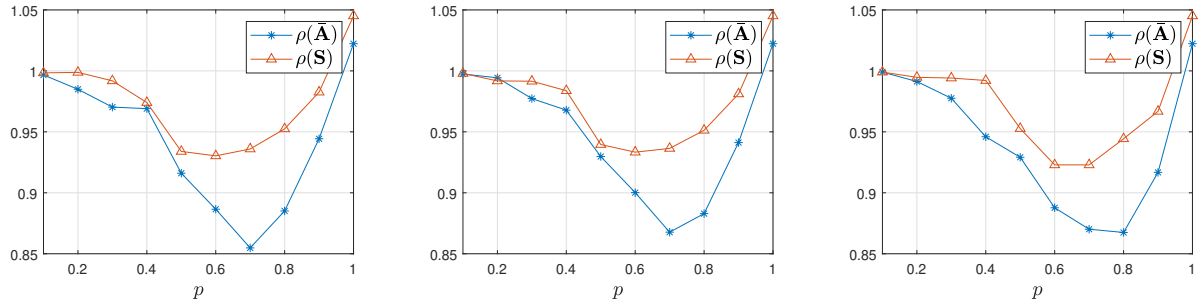
Figure 4.6: Mean 100 experiments of the convergence rate of asynchronous affine fixed point problem for different values of $\rho(\bar{\mathbf{A}})$ and for different values of $\rho(\mathbf{S})$, for two different matrices $\mathbf{P}$.

4.1.2.3 A Solution to the Problem

The problem we want to solve is that of minimizing the convergence time of the affine updates of asynchronous scheduling. As we have seen, to do that we can minimize the spectral radius, $\rho(\bar{\mathbf{A}})$. Thus, we want to find the proper combination of probabilities of update, for each cluster of nodes, such that we get the minimal $\rho(\bar{\mathbf{A}})$.

We can approximate the solution of this problem experimentally. To do this, we get possible combinations of probabilities of update, based on the number of clusters we have. After that we compute $\rho(\bar{\mathbf{A}})$, and find the combination of probabilities that minimizes it. In Figures 4.9, 4.10, we can see the values of $\rho(\bar{\mathbf{A}})$ and $\rho(\mathbf{S})$, for the matrix $\mathbf{A}$ that is generated using matrices $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$ and $\mathbf{M}_4$, as they are defined in Appendix A, when GBP is used to solve a linear system of equations. All possible combinations of probabilities of update are examined, using the values in vector $[0.1 : 0.1 : 1]$, for different values of k. Also, in Figures 4.7, 4.8, we can see similar results but for terminals that have equal probabilities, to get a clearer result. As we can see the minima of $\rho(\bar{\mathbf{A}})$ and $\rho(\mathbf{S})$ are different in some cases, but mostly close to each other.

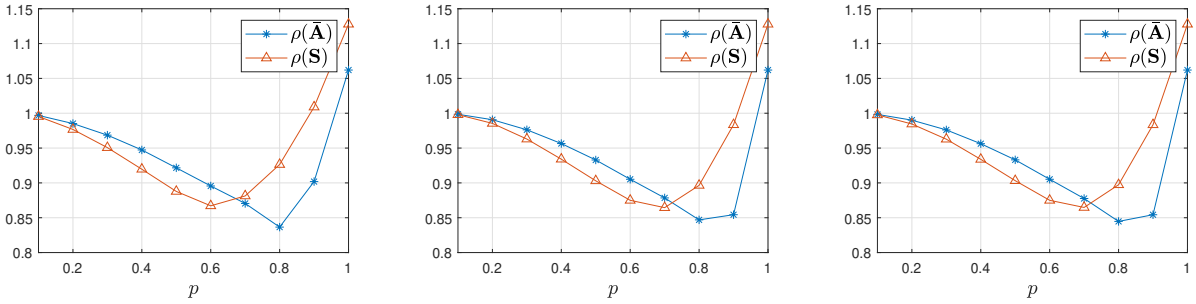
In all of the experiments performed, first we clustered the generated graph using spectral clustering. The clustering is required to compute the probabilities of update for each edge of the graph, which are used in matrix $\mathbf{P}$.



(a1) $k = 2$ (a2) $k = 3$ (a3) $k = 4$

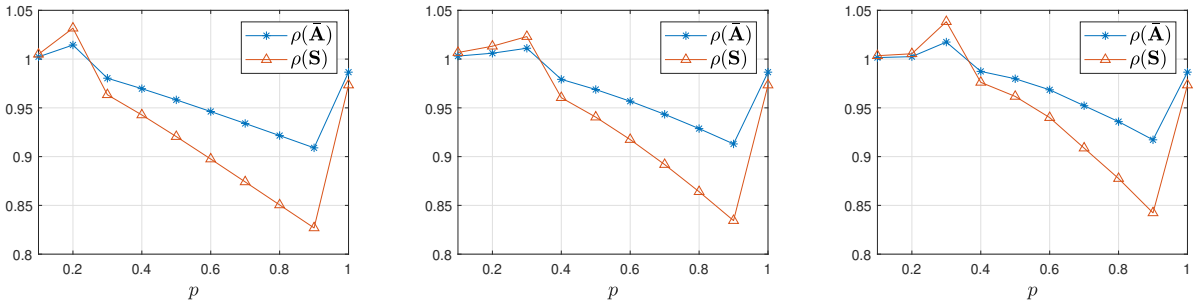
(a) $\mathbf{M}_1$

Figure 4.7: Experimental computation of $\rho(\bar{\mathbf{A}})$ and $\rho(\mathbf{S})$, for the matrix $\mathbf{A}$ that is generated using matrix $\mathbf{M}_1$. Here we present the results only for clusters with equal probabilities of update, using the values in vector $[0.1 : 0.1 : 1]$, for different values of k.



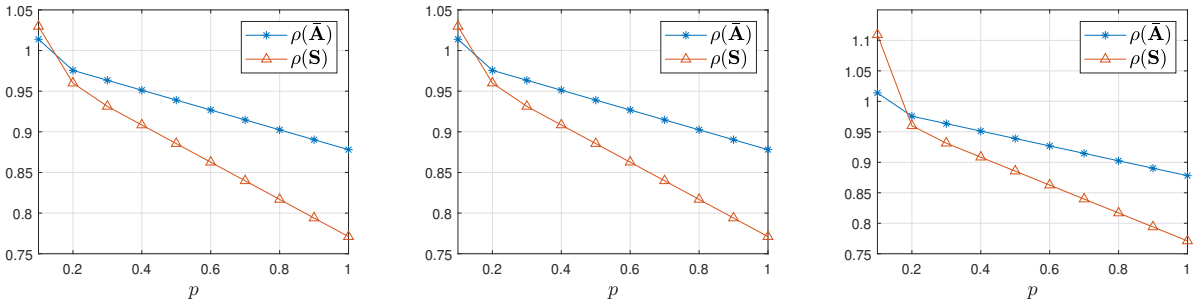
(a1) $k = 2$ (a2) $k = 3$ (a3) $k = 4$

(a) $\mathbf{M}_2$



(b1) $k = 2$ (b2) $k = 3$ (b3) $k = 4$

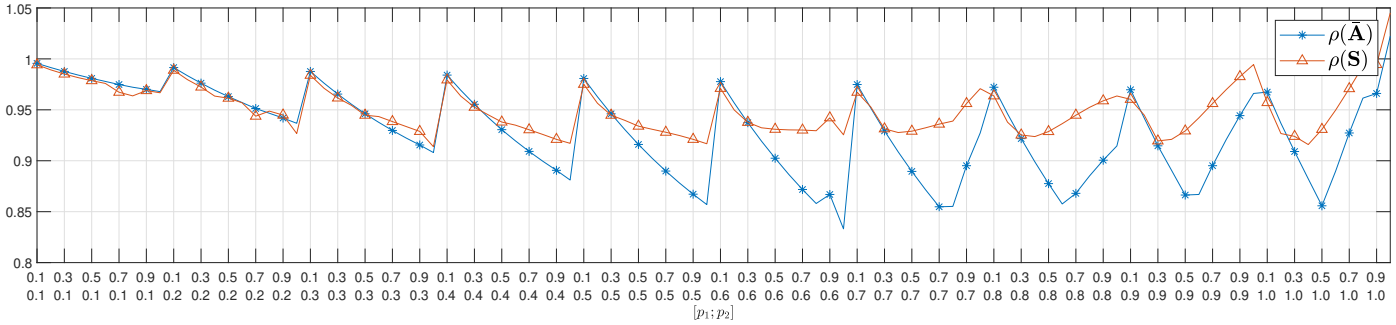
(b) $\mathbf{M}_3$



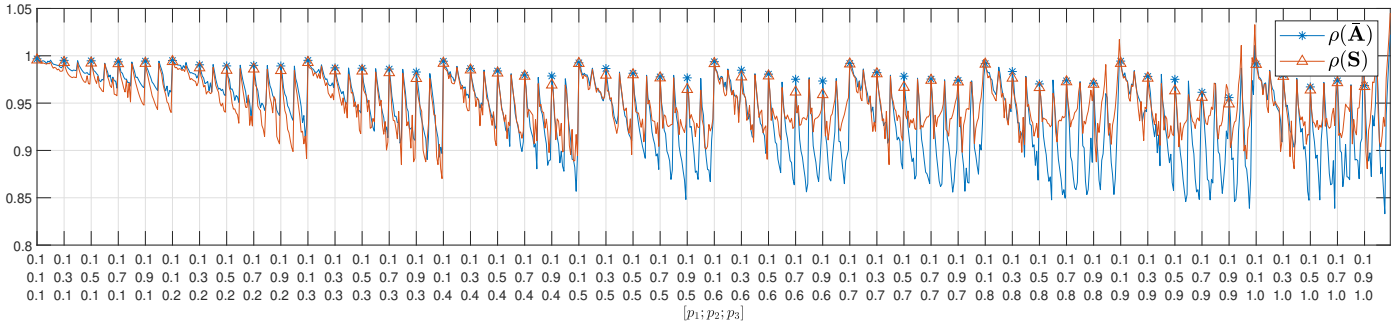
(c1) $k = 2$ (c2) $k = 3$ (c3) $k = 4$

(c) $\mathbf{M}_4$

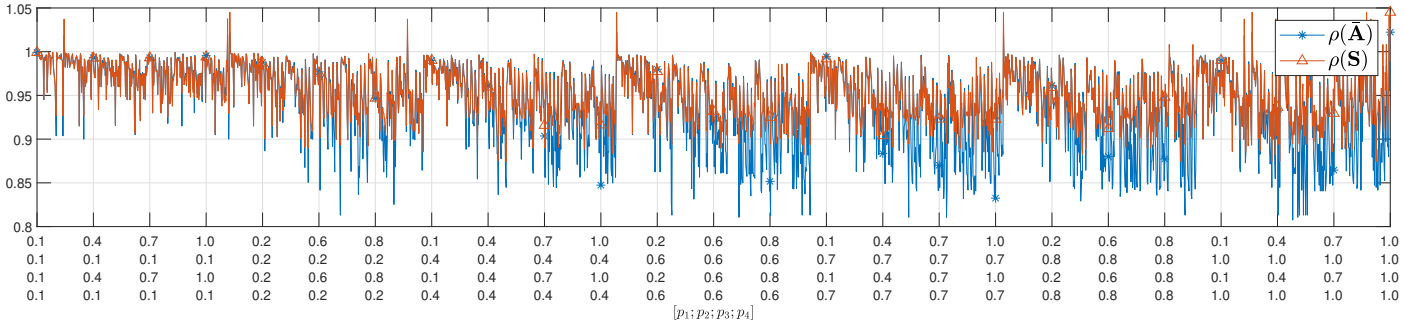
Figure 4.8: Experimental computation of $\rho(\bar{\mathbf{A}})$ and $\rho(\mathbf{S})$, for the matrix $\mathbf{A}$ that is generated using matrices $\mathbf{M}_2$, $\mathbf{M}_3$ and $\mathbf{M}_4$. Here we present the results only for clusters with equal probabilities of update, using the values in vector $[0.1 : 0.1 : 1]$, for different values of k.



(a) $k = 2$

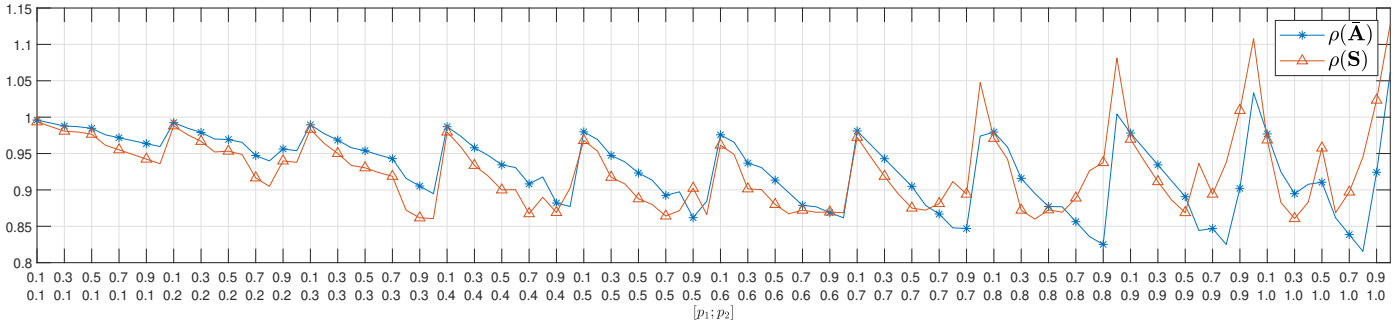


(b) $k = 3$

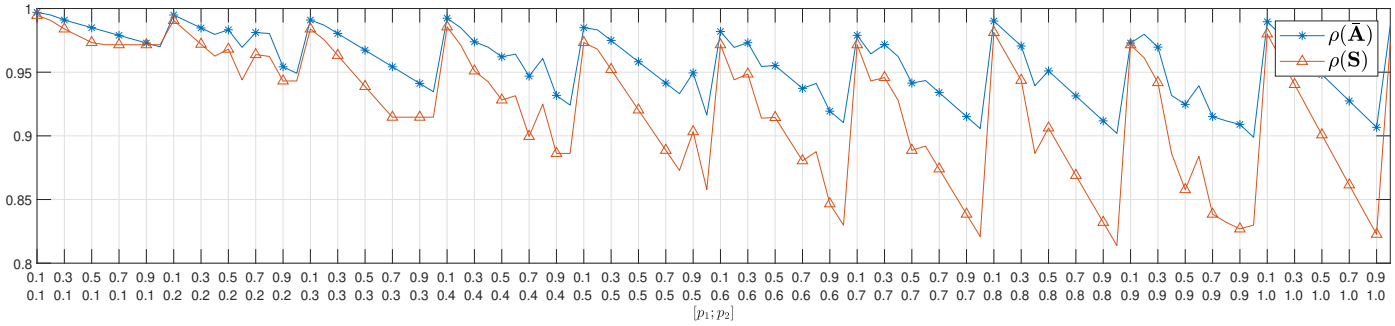


(c) $k = 4$

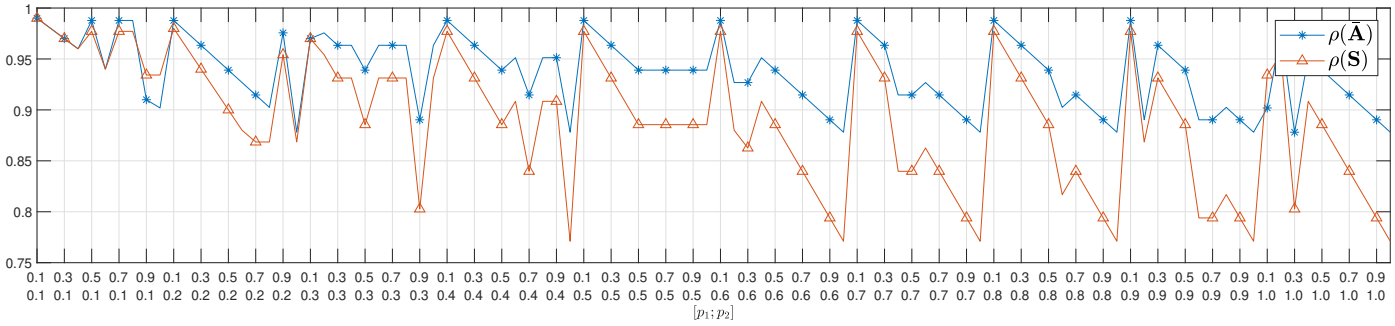
Figure 4.9: Experimental computation of $\rho(\bar{\mathbf{A}})$ and $\rho(\mathbf{S})$, for the matrix $\mathbf{A}$ that is generated using the matrix $\mathbf{M}_1$ and possible combinations of probabilities of update. All possible combinations of probabilities of update are examined, using the values in vector $[0.1 : 0.1 : 1]$, for different values of k.



(a) $\mathbf{M}_2$



(b) $\mathbf{M}_3$



(c) $\mathbf{M}_4$

Figure 4.10: Experimental computation of $\rho(\bar{\mathbf{A}})$ and $\rho(\mathbf{S})$, for the matrix $\mathbf{A}$ that is generated using matrice $\mathbf{M}_2$, $\mathbf{M}_3$ and $\mathbf{M}_4$. Here all possible combinations of probabilities of update are examined, using the values in vector $[0.1 : 0.1 : 1]$, for $k = 2$ clusters.

4.1.2.4 Simulations with Gaussian Belief Propagation

In this section, we present for one more time experimental results of the convergence rate of Gaussian Belief Propagation when it is utilized to solve linear systems of equations. We want to compare its performance when the produced graph that GBP uses, is clustered using random clustering, k-means, spectral clustering and autonomous clustering.

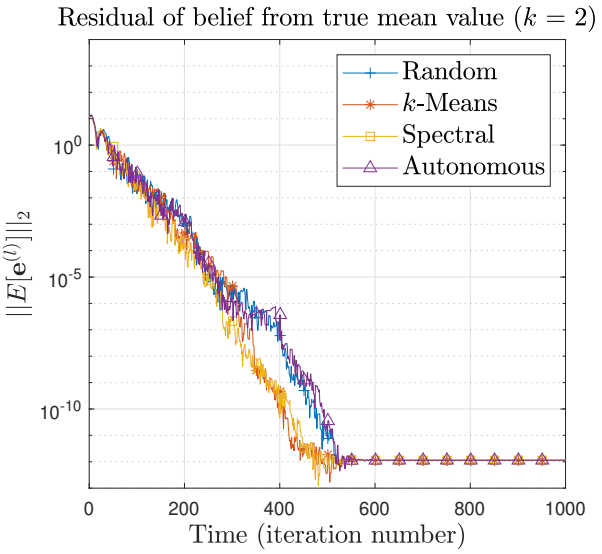
We present experimental results for GBP solving a linear system of equations in an asynchronous manner. In particular, we plot the estimation of the per iteration expected value $\|\mathbb{E}[\mathbf{e}^{(l)}]\|_2$ using 20 independent experiments, with the error $\mathbf{e}$ as defined in Equation 4.4.

In Figures 4.11, 4.12, 4.13 and 4.14 we present results for random clustering, k-means, spectral clustering and autonomous clustering when we have 2, 3 or 4 WSN terminals. We used as probabilities of update of each WSN terminal, the experimental values of the optimal probabilities as they can be seen in Figures 4.7 and 4.8, in the previous section. The values that are used in our experiments are presented in Table 4.1.

	$k=2$	$k=3$	$k=4$
$\mathbf{M}_1$	0.7	0.7	0.8
$\mathbf{M}_2$	0.8	0.8	0.8
$\mathbf{M}_3$	0.9	0.9	0.9
$\mathbf{M}_4$	1	1	1

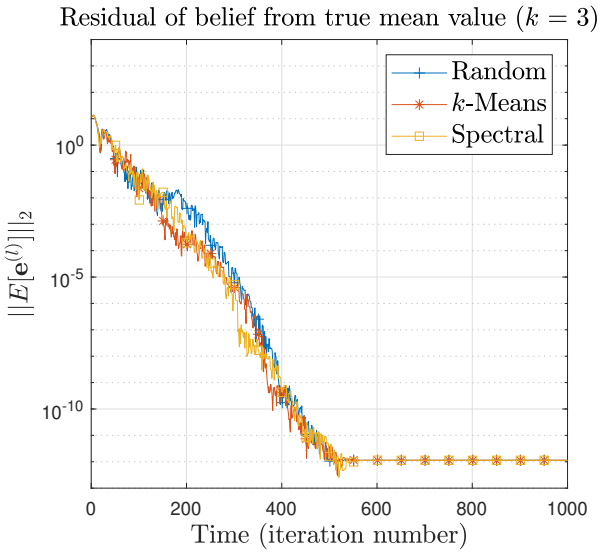
Table 4.1: Optimal probabilities of update, as obtained from experimental results in Figures 4.7 and 4.8.

We can see in Figures 4.11, 4.12, 4.13 and 4.14 that for the optimal probabilities of update, we have faster convergence than we had before with $p = 0.5$ for all terminals. The amazing result here is that for those probabilities we have about the same convergence rate even in the case of *random* clustering! Again, this is something that needs further theoretical examination.



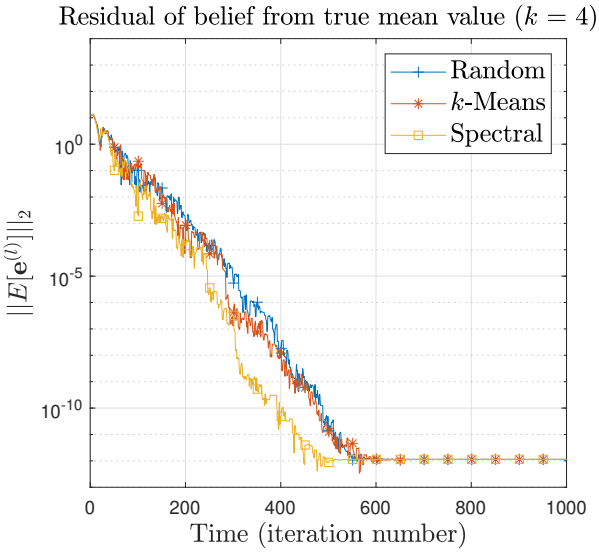
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.866996,\ \rho(\mathbf{S}) = 0.933406$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.854809,\ \rho(\mathbf{S}) = 0.935944$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.854809,\ \rho(\mathbf{S}) = 0.935944$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.854809,\ \rho(\mathbf{S}) = 0.935944$



(b)

Random	$\rho(\bar{\mathbf{A}}) = 0.870251,\ \rho(\mathbf{S}) = 0.931730$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.868430,\ \rho(\mathbf{S}) = 0.934825$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.867740,\ \rho(\mathbf{S}) = 0.936418$

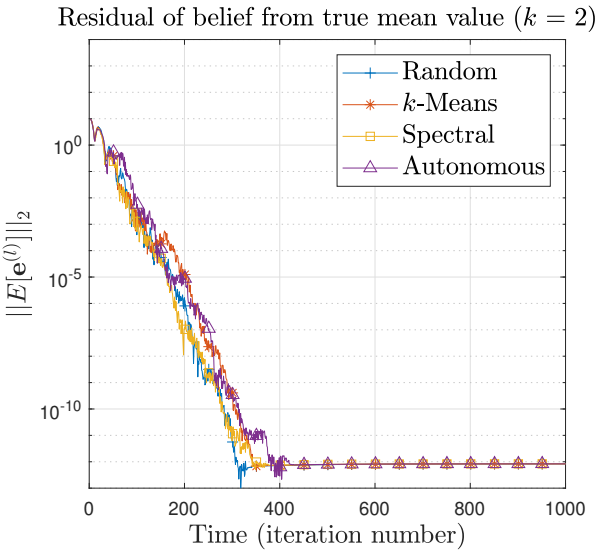


(c)

Random	$\rho(\bar{\mathbf{A}}) = 0.883246,\ \rho(\mathbf{S}) = 0.932321$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.878339,\ \rho(\mathbf{S}) = 0.932046$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.861155,\ \rho(\mathbf{S}) = 0.924399$

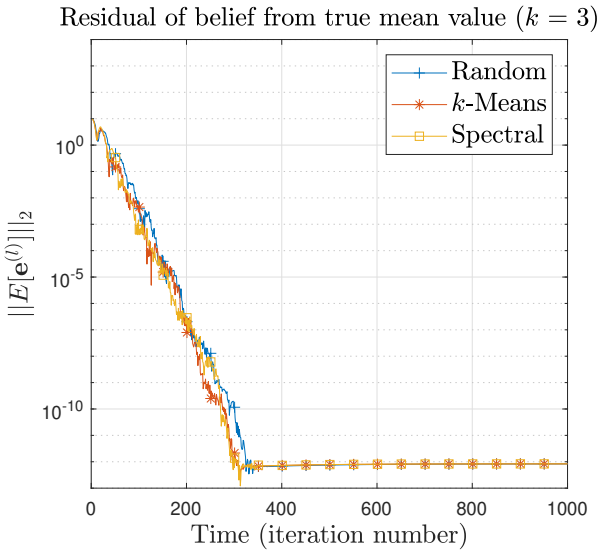
Figure 4.11: $\mathbf{M}_1$

Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to the optimal probabilities that can be found in Figures 4.7 and 4.8.



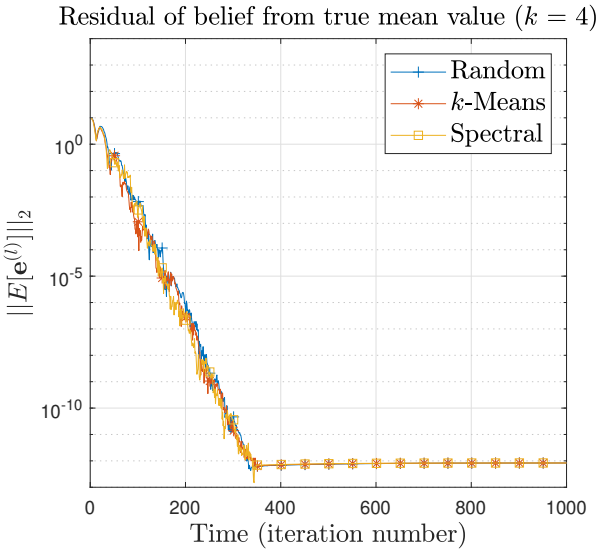
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.842130,\ \rho(\mathbf{S}) = 0.888642$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.838606,\ \rho(\mathbf{S}) = 0.903156$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.836695,\ \rho(\mathbf{S}) = 0.926447$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.835668,\ \rho(\mathbf{S}) = 0.926408$



(b)

Random	$\rho(\bar{\mathbf{A}}) = 0.858757,\ \rho(\mathbf{S}) = 0.873433$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.845201,\ \rho(\mathbf{S}) = 0.893528$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.844888,\ \rho(\mathbf{S}) = 0.897441$

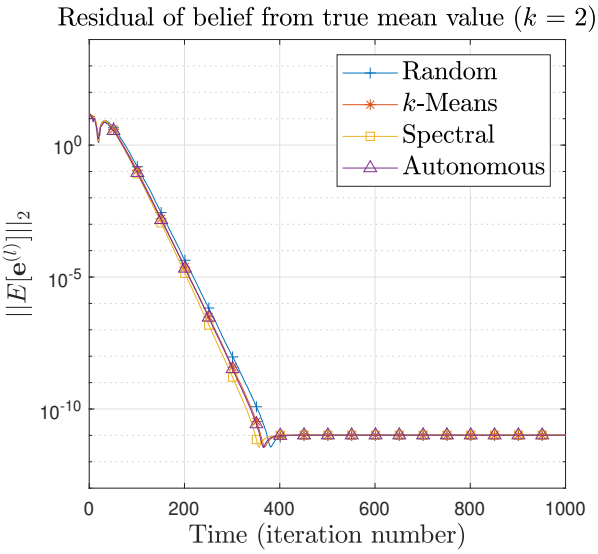


(c)

Random	$\rho(\bar{\mathbf{A}}) = 0.856639,\ \rho(\mathbf{S}) = 0.873246$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.853057,\ \rho(\mathbf{S}) = 0.883966$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.843889,\ \rho(\mathbf{S}) = 0.897730$

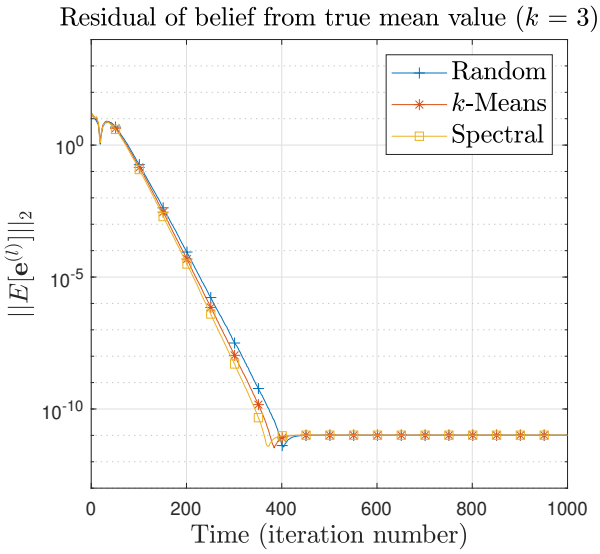
Figure 4.12: $\mathbf{M}_2$

Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to the optimal probabilities that can be found in Figures 4.7 and 4.8.



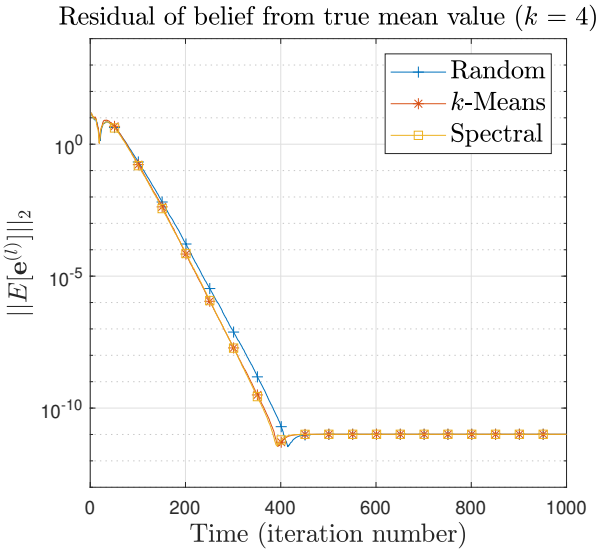
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.914313,\ \rho(\mathbf{S}) = 0.836896$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.912199,\ \rho(\mathbf{S}) = 0.832988$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.908991,\ \rho(\mathbf{S}) = 0.826900$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.911839,\ \rho(\mathbf{S}) = 0.832308$



(b)

Random	$\rho(\bar{\mathbf{A}}) = 0.917722,\ \rho(\mathbf{S}) = 0.843326$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.914735,\ \rho(\mathbf{S}) = 0.837693$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.913003,\ \rho(\mathbf{S}) = 0.834404$

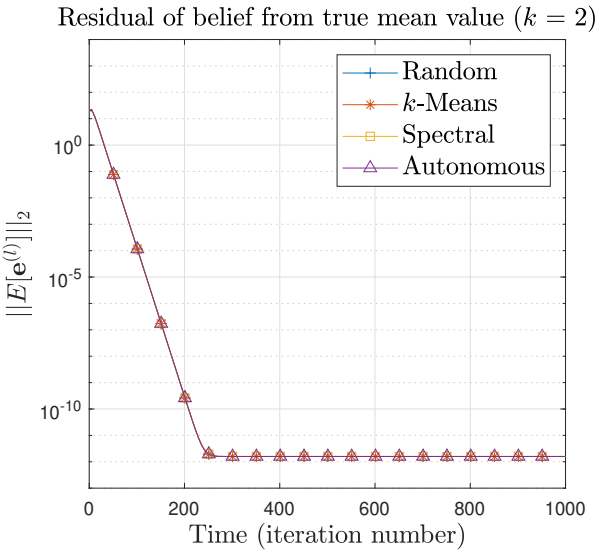


(c)

Random	$\rho(\bar{\mathbf{A}}) = 0.920076,\ \rho(\mathbf{S}) = 0.847838$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.916755,\ \rho(\mathbf{S}) = 0.841499$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.917227,\ \rho(\mathbf{S}) = 0.842327$

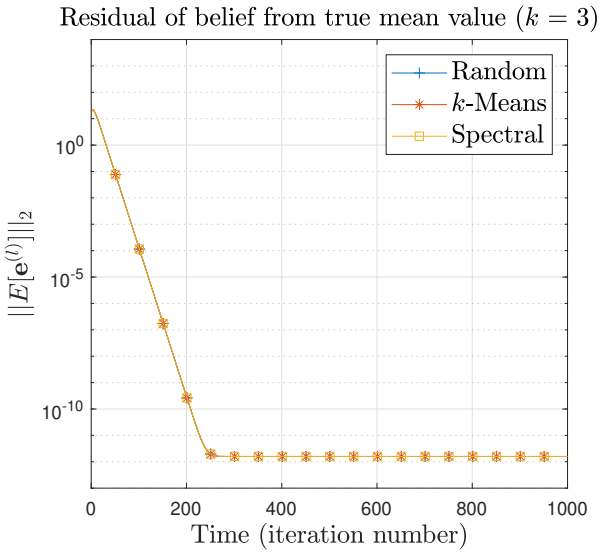
Figure 4.13: $\mathbf{M}_3$

Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to the optimal probabilities that can be found in Figures 4.7 and 4.8.



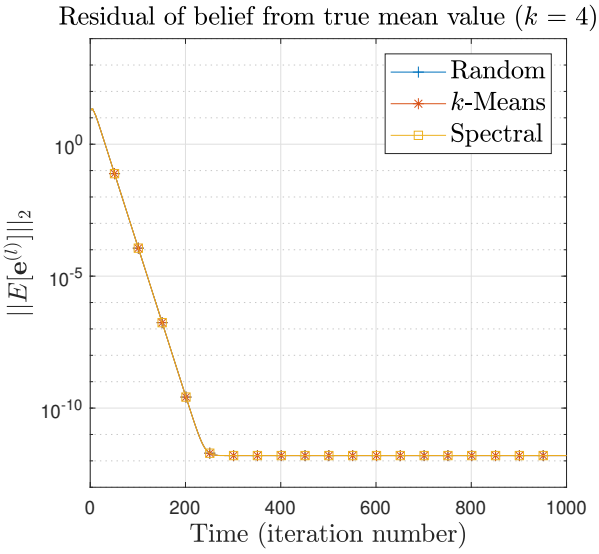
(a)

Random	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
Autonomous	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$



(b)

Random	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$



(c)

Random	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
k-Means	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$
Spectral	$\rho(\bar{\mathbf{A}}) = 0.878088,\ \rho(\mathbf{S}) = 0.771039$

Figure 4.14: $\mathbf{M}_4$

Convergence of asynchronous Gaussian Belief Propagation for different clustering methods and different number of clusters. In particular, we present results for WSNs with terminals which have probabilities of update equal to the optimal probabilities that can be found in Figures 4.7 and 4.8.

4.2 Minimization of Energy Consumption

We can define the following optimization problem

$$\begin{aligned} \min \quad & \text{Energy Consumption} \\ \text{s.t.} \quad & \rho(\bar{\mathbf{A}}) < 1 \quad \text{and} \quad \rho(\mathbf{S}) < 1 \end{aligned} \tag{4.7}$$

Each WSN terminal consists of a Silab's Thunderboard Sense 2 (EFR32MG12 IC) embedded module, and according to previous work of our group [35], the cost of each operation (sum/product) is much smaller than the communication cost:

- Operation Cost $\approx 1\, nJoule$,
- Communication Cost $\approx 6.5\, mJoule$.

The number of operations that are being done are almost the same for every run of the algorithm (almost, because they depend on the random quantity $\mathbf{\Psi}$) and independent of the clustering. On the other hand, the number of messages that are being sent between WSN terminals are different for each clustering and depend on the number of edges that connect the WSN terminals. Thus, we should minimize the *Communication* Energy Consumption, i.e. the number of edges that connect the WSN terminals, which is minimized by minimizing the *RatioCut*.

Hence, problem 4.7 is equivalent to

$$\begin{aligned} \min \quad & \text{Communication Energy Consumption} \\ \text{s.t.} \quad & \rho(\bar{\mathbf{A}}) < 1 \quad \text{and} \quad \rho(\mathbf{S}) < 1 \end{aligned}, \tag{4.8}$$

which is equivalent to

$$\begin{aligned} \min \quad & \text{Connecting Edges} \\ \text{s.t.} \quad & \rho(\bar{\mathbf{A}}) < 1 \quad \text{and} \quad \rho(\mathbf{S}) < 1 \end{aligned}, \tag{4.9}$$

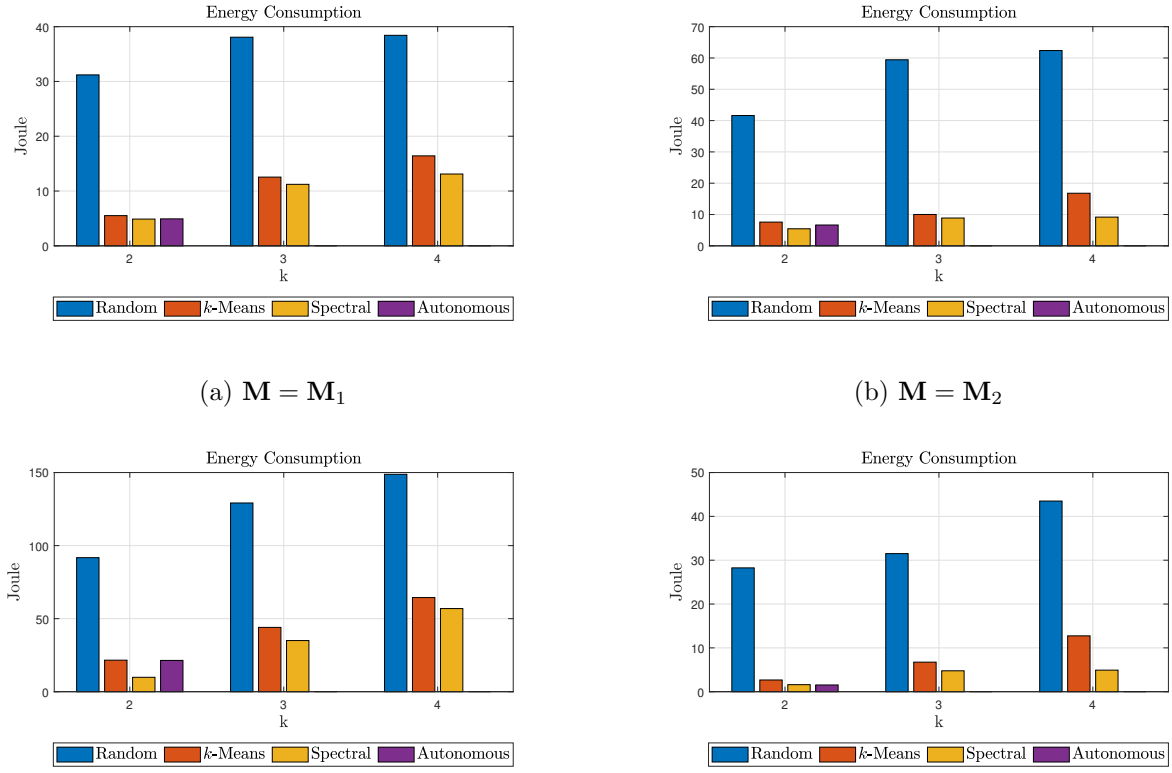
which is equivalent to

$$\begin{aligned} \min \quad & \textit{RatioCut} \\ \text{s.t.} \quad & \rho(\bar{\mathbf{A}}) < 1 \quad \text{and} \quad \rho(\mathbf{S}) < 1 \end{aligned}. \tag{4.10}$$

As we saw in Section 3.2, the minimization of *RatioCut* is a problem that can be solved using spectral clustering. In our problem we have also the constraint of $\rho(\bar{\mathbf{A}}) < 1$ and $\rho(\mathbf{S}) < 1$. Thus, to use spectral clustering, we can relax our problem by removing the constraints. Then, we have two possible cases for the probabilities of update of each WSN terminal.

- We can have fixed probabilities for each terminal. In that case, after the clustering is performed, we can just check if the constraints $\rho(\bar{\mathbf{A}}) < 1$ and $\rho(\mathbf{S}) < 1$, are fulfilled.
- We might not have fixed probabilities for each terminal. Then we have to set their values. In that case, after the clustering is performed, we can select a set of probabilities which suits our constraints, as we are going to see in Section 4.1.

We want to see the total energy that is consumed by GBP in solving linear systems of equations, until we reach a certain threshold in the error between the least squares solution of the system and the computed solution. We run 20 experiments, with equal probabilities of update for all terminals ($p = 0.5$), to make it comparable for the different number of clusters, $k = 2, 3, 4$, and threshold $= 10^{-5}$. In Figure 4.15 we can see the mean value of energy consumption for random clustering, k-means, spectral clustering and autonomous clustering.



(a) $\mathbf{M} = \mathbf{M}_1$

(b) $\mathbf{M} = \mathbf{M}_2$

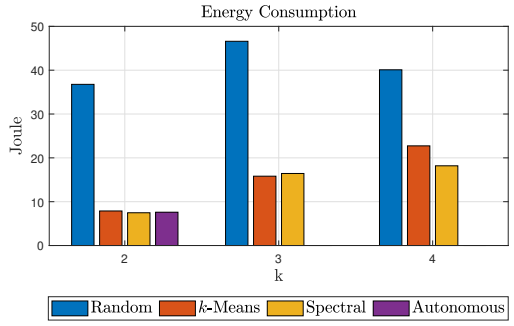
(c) $\mathbf{M} = \mathbf{M}_3$

(d) $\mathbf{M} = \mathbf{M}_4$

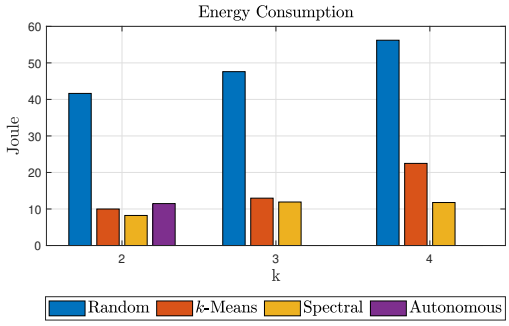
Figure 4.15: Experimental computation of energy that is consumed by GBP in solving a linear system of equations $\mathbf{Mx} = \mathbf{s}$. Specifically, we run 20 experiments, with equal probabilities of update for all terminals ($p = 0.5$), for $k = 2, 3, 4$, threshold $= 10^{-5}$ and for 4 different matrices $\mathbf{M}$.

Afterwards, we used the probabilities that minimize $\rho(\bar{\mathbf{A}})$, which can be found from Figures 4.7 and 4.8. Again, we run 20 experiments, for $k = 2, 3, 4$ and threshold $= 10^{-5}$. In Figure 4.16 we can see the mean value of energy consumption for random clustering, k-means, spectral clustering and autonomous clustering.

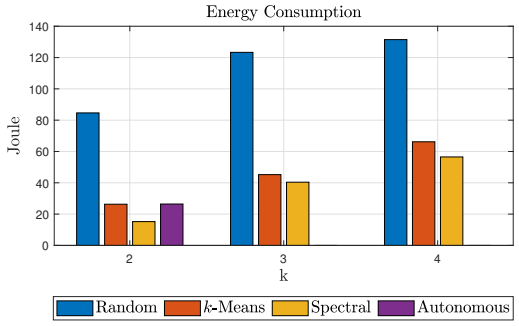
As we can see in Figures 4.15 and 4.16, in most cases, when the number of clusters gets bigger, the energy consumption also increases. We can see that for spectral clustering, for $\mathbf{M}_2$ and $\mathbf{M}_4$ energy consumption is almost equal between $k = 3$ and $k = 4$. Also, in most experiments spectral clustering performs better than the others, and autonomous clustering has a performance close to both k-means and spectral clustering.



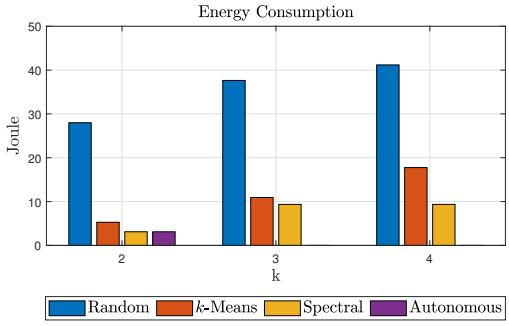
(a) $\mathbf{M} = \mathbf{M}_1$



(b) $\mathbf{M} = \mathbf{M}_2$



(c) $\mathbf{M} = \mathbf{M}_3$



(d) $\mathbf{M} = \mathbf{M}_4$

Figure 4.16: Experimental computation of energy that is consumed by GBP in solving a linear system of equations $\mathbf{Mx} = \mathbf{s}$. Specifically, we run 20 experiments, with the optimal probabilities of update, as they can be found from Figures 4.7 and 4.8, for $k = 2, 3, 4$, threshold $= 10^{-5}$ and for 4 different matrices $\mathbf{M}$.

Chapter 5

Conclusions and Future Work

In this work our goal is to map the Probabilistic Graphical Models (PGM) to Wireless Sensor Networks' (WSN) terminals, in order to solve the affine fixed point problem

$$\mathbf{x}^{(l)} = \mathbf{A}\mathbf{x}^{(l-1)} + \mathbf{b} \tag{5.1}$$

in a distributed manner, without any cloud/edge computing support. Since WSNs may not have sufficient power for the computations and some nodes may fail to operate, *asynchronous* scheduling needs to be utilized.

We considered a famous inference algorithm, *Gaussian Belief Propagation (GBP)*, which can be modeled using Equation 5.1. We used this algorithm to solve linear systems of equations. To do so in an *asynchronous* manner and in form of an actual WSN, we had to map the produced graph to the nodes of the network. Thus, we presented two different approaches of clustering; the *edge clustering* and the *node clustering*, with the latter prevailing. Afterwards, we presented an autonomous, in-network clustering that reaches the performance of k-means and spectral clustering. Furthermore, we showed that non-random mappings achieve better performance in terms of energy consumption, especially when the PGM is clustered by the spectral clustering algorithm. Finally, we provided a strong relation between spectral radius and convergence rate and showed that when we choose the right set of probabilities of update, for the WSN terminals, we achieve the fastest convergence to the fixed point, independently of the clustering!

One interesting future direction of this work is to generalize the autonomous clustering algorithm to more than 2 clusters and also to find a way to compute the *algebraic connectivity* λ_2 in an asynchronous manner. Another direction could be to integrate the matrix $\mathbf{A}$ in the clustering procedure, as the affine updates are based on this matrix and its non-zero elements. In addition, more inference algorithms could be studied such as Kalman filtering, which can be viewed as a special case of GBP in Hidden Markov Networks (HMNs), or the Viterbi algorithm for sequence detection in communication problems, which can be viewed as a special case of max-product in HMNs, or Expectation-Maximization and many more.

Appendix A

Matrix M

In this Appendix we present the matrices $\mathbf{M}$ that were used in our experiments, in order to solve the linear system of equations $\mathbf{Mx} = \mathbf{s}$, with $\mathbf{s} = \mathbf{1}$ (all one vector). Also, in Figures A.1, A.2, A.3 and A.4 we can see the graph that is being generated from each matrix.

$$\mathbf{M}_1 = \begin{bmatrix} 3.63 & -6.12 & 0 & 0 & 0 & -2.61 & 0 & 0 \\ 0 & -10.65 & 7.59 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.92 & 7.05 & 0 & 0 & -10.46 & 0 \\ 0 & 0 & 0 & 0.18 & 3.27 & 0 & 0 & 0 \\ -2.01 & 0 & 0 & 0 & -0.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8.01 & 0.37 \\ 5.18 & -1.86 & 0 & 4.63 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.91 & 0 & 0 & 0 & 0 & 0.13 \end{bmatrix} \tag{A.1}$$

$$\mathbf{M}_2 = \begin{bmatrix} -1.7641 & -0.1624 & 0 & 0 & 0 & 0 & 0 & 1.5053 \\ 0.7257 & -0.5360 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.1374 & 0.6552 & 1.3097 & 0 & 0 & 0 \\ 0.8566 & 0 & 1.7936 & -0.5378 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.3709 & 0 & -0.9888 & 0.0055 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2486 & 0 & -0.4413 \\ 0 & 0 & 0 & 0 & 0 & -1.1012 & -1.9016 & -0.7928 \\ 0 & 0 & 0 & 0 & 0 & 0.0034 & 0 & -0.8377 \end{bmatrix} \tag{A.2}$$

$$\mathbf{M}_3 = \begin{bmatrix} 0.8305 & -1.0524 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.5982 & -0.5424 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.4126 & -1.7550 & 0 & 0 & 0 & -0.5872 & 0 \\ 0 & 0 & 1.7936 & -0.5378 & 0 & 0 & 0 & 0 \\ 0 & -0.8435 & -0.7876 & 0.2653 & 0 & 0 & 0.2715 & 0 \\ 0 & 0 & 0 & 1.1617 & 0 & -0.3724 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7369 & 0.2165 & -1.4714 & 0.1944 \\ 0 & 0 & 0 & 0 & 0 & -0.6676 & 0 & 0.9759 \end{bmatrix} \tag{A.3}$$

$$\mathbf{M}_4 = \begin{bmatrix} 0 & 0 & 0.7895 & 0 & 0 & -0.7525 & 0 & -0.2518 \\ 0 & 0 & 0 & -1.3945 & 1.4289 & 0 & 0 & 0 \\ -0.6963 & 0 & 0 & 0 & 0 & 0 & 0 & 1.8027 \\ 0 & 1.8073 & 1.3534 & 0 & 0 & -1.3617 & -1.6536 & 0 \\ 0.6424 & 0 & 0 & 0 & 0 & 0 & 0 & -1.6058 \\ 0 & 0 & 0 & 0 & 0 & -1.1537 & -0.2274 & -0.2837 \\ 0 & 0 & 0.2340 & 0.8620 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2491 & -1.2132 & 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

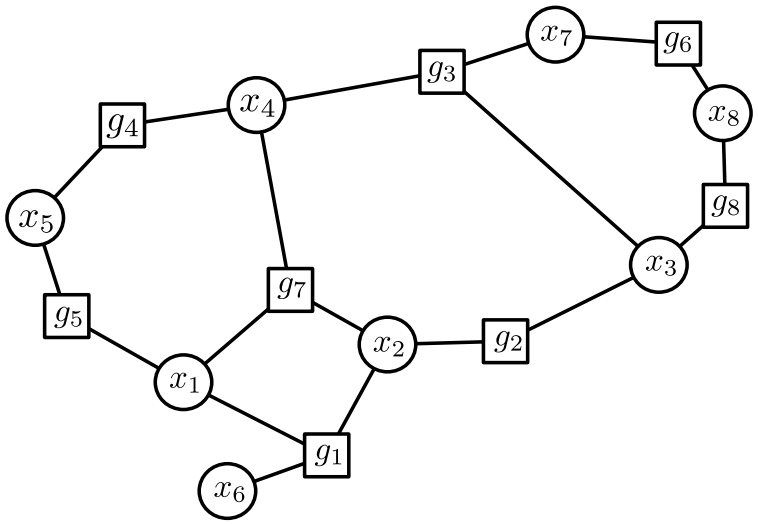


Figure A.1: $\mathbf{M}_1$

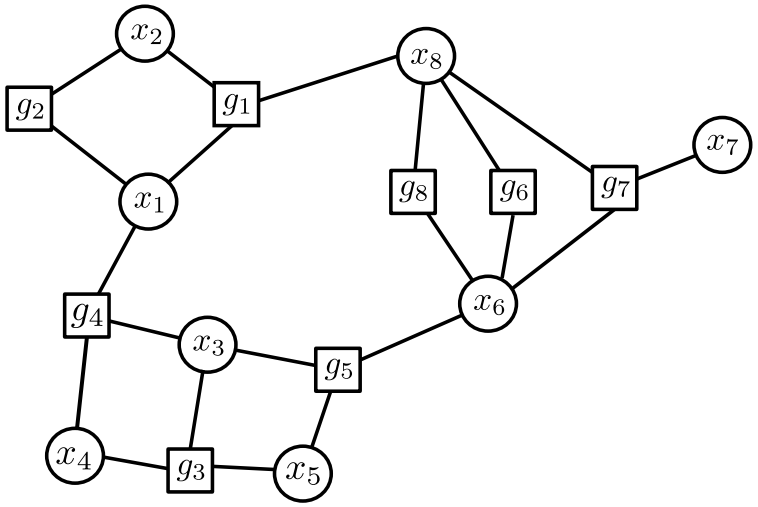


Figure A.2: $\mathbf{M}_2$

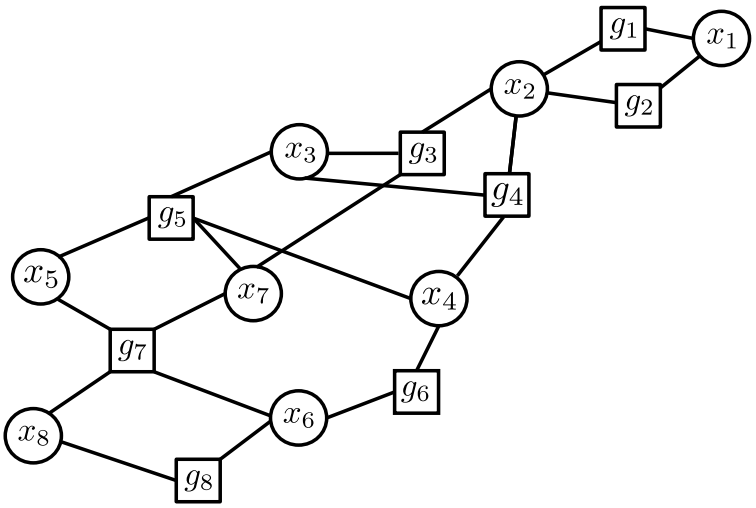


Figure A.3: $\mathbf{M}_3$

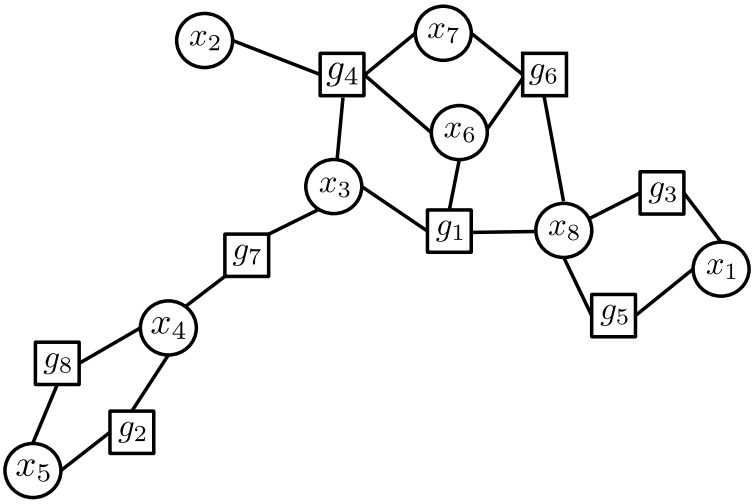


Figure A.4: $\mathbf{M}_4$

Appendix B

Proofs

B.1 Proof of Theorem 3.1

(The proof is based on [26].)
We start with the case of $k = 1$, that is the graph is connected. Assume that $\mathbf{f}$ is an eigenvector with eigenvalue 0. Then we know that,

$$0 = \mathbf{f}^\top \mathbf{L}\mathbf{f} = \sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2.$$

As the weights w_{ij} are non-negative, this sum can only vanish if all terms $w_{ij}(f_i - f_j)^2$ are equal to zero. Thus, if two vertices v_i and v_j are connected (i.e., $w_{ij} > 0$), then f_i needs to equal to f_j. With this argument we can see that $\mathbf{f}$ needs to be constant for all vertices which can be connected by a path in the graph. Moreover, as all vertices of a connected component in an undirected graph can be connected by a path, $\mathbf{f}$ needs to be constant on the whole connected component. In a graph consisting of only one connected component, we only have the constant one vector $\mathbf{1}$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.

Now consider the case of k connected components. Without loss of generality we assume that the vertices are ordered according to the connected components they belong to. In this case, the weight matrix $\mathbf{W}$ has a block diagonal form, and the same is true for the matrix $\mathbf{L}$:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{L}_k \end{bmatrix}.$$

Note that each of the blocks $\mathbf{L}_i$ is a proper graph Laplacian on its own, namely the Laplacian corresponding to the subgraph of the i-th connected component. As it is the case for all block diagonal matrices, we know that the spectrum of $\mathbf{L}$ is given by the union of the spectra of $\mathbf{L}_i$, and

that the corresponding eigenvectors of $\mathbf{L}$ are the eigenvectors of $\mathbf{L}_i$, filled with 0 at the positions of the other blocks. As each $\mathbf{L}_i$ is a graph Laplacian of a connected graph, we know that every $\mathbf{L}_i$ has eigenvalue 0 with multiplicity 1, and the corresponding eigenvector is the constant one vector on the i-th connected component. Thus, the matrix $\mathbf{L}$ has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components.

B.2 Proof of Proposition 3.1

(The proof is based on [26].)

(1) By the definition of d_i,

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{D}\mathbf{f} - \mathbf{f}^\top \mathbf{W}\mathbf{f} = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{i=1}^{n} d_i f_i^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

(2) $\mathbf{L}$ is symmetric as the sum of 2 symmetric matrices and positive semi-definite as we shown, for Property (1), that $\mathbf{f}^\top \mathbf{L}\mathbf{f} \geq 0,\ \forall \mathbf{f} \in \mathbb{R}^n$.

(3) If $\mathbf{L}$ has an eigendecomposition $\mathbf{L}\mathbf{V} = \mathbf{V}\mathbf{\Lambda}$, columns in $\mathbf{V}$ are eigenvectors, and $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues. We can easily rewrite the eigendecomposition as $\mathbf{V}^\top \mathbf{L}\mathbf{V} = \mathbf{\Lambda}$, with each element of $\mathbf{\Lambda}$, $\lambda_i = v_i^\top \mathbf{L} v_i \Rightarrow \lambda_i = v_i^\top \mathbf{L} v_i \geq 0$, because $\mathbf{L}$ is positive semi-definite.

Because the sum of all rows of $\mathbf{L}$ is 0 (Equation 3.2), the rows are linearly dependent, thus the matrix is not full rank. So, there should be at least one eigenvalue that is 0. Also, from Equation 3.2, it follows that the corresponding eigenvector is the constant one vector, $\mathbf{1}$.

(4) A direct consequence of Properties (1) - (3).

B.3 Proof of Proposition 3.2

(The proof is based on [36].)

(1) We proved that

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}(f_i - f_j)^2,$$

thus,

$$\mathbf{f}_{\mathcal{A}}^\top \mathbf{L}\mathbf{f}_{\mathcal{A}} = \frac{1}{2}\sum_{i,j=1}^{n} w_{i,j}((\mathbf{f}_{\mathcal{A}})_i - (\mathbf{f}_{\mathcal{A}})_j)^2,$$

It is clear that, when $v_i, v_j \in \mathcal{A}$ or $v_i, v_j \in \bar{\mathcal{A}}$, it holds that $(\mathbf{f}_{\mathcal{A}})_i - (\mathbf{f}_{\mathcal{A}})_j = 0$.

So we can write,

$$
\begin{aligned}
\mathbf{f}_{\mathcal{A}}^{\top}\mathbf{L}\mathbf{f}_{\mathcal{A}} &= \frac{1}{2}\sum_{i,j\,:\,v_i\in\mathcal{A},v_j\in\bar{\mathcal{A}}}^{n} w_{i,j}((\mathbf{f}_{\mathcal{A}})_i - (\mathbf{f}_{\mathcal{A}})_j)^2 + \frac{1}{2}\sum_{i,j\,:\,v_i\in\bar{\mathcal{A}},v_j\in\mathcal{A}}^{n} w_{i,j}((\mathbf{f}_{\mathcal{A}})_i - (\mathbf{f}_{\mathcal{A}})_j)^2 \\
&= \frac{1}{2}\sum_{i,j\,:\,v_i\in\mathcal{A},v_j\in\bar{\mathcal{A}}}^{n} w_{i,j}\left(\sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} + \sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}}\right)^2 + \frac{1}{2}\sum_{i,j\,:\,v_i\in\bar{\mathcal{A}},v_j\in\mathcal{A}}^{n} w_{i,j}\left(-\sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}} - \sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}}\right)^2 \\
&= \frac{1}{2}\left(\sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} + \sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}}\right)^2 \cdot cut(\mathcal{A}) + \frac{1}{2}\left(-\sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}} - \sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}}\right)^2 \cdot (\bar{\mathcal{A}}) \\
&= cut(\mathcal{A}) \cdot \left(\sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} + \sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}}\right)^2, \quad \left[cut(\mathcal{A}) = cut(\bar{\mathcal{A}})\right] \\
&= cut(\mathcal{A}) \cdot \left(\sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}}^2 + 2 \cdot \sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} \cdot \sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}} + \sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}}^2\right) \\
&= cut(\mathcal{A}) \cdot \left(\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|} + 2 + \frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}\right) = cut(\mathcal{A}) \cdot \left(\frac{n - |\mathcal{A}|}{|\mathcal{A}|} + \frac{|\mathcal{A}| + n - |\mathcal{A}|}{|\bar{\mathcal{A}}|} + 1\right) \\
&= cut(\mathcal{A}) \cdot \left(\frac{n}{|\mathcal{A}|} - 1 + \frac{n}{|\bar{\mathcal{A}}|} + 1\right) = n \cdot cut(\mathcal{A})\left(\frac{1}{|\mathcal{A}|} + \frac{1}{|\bar{\mathcal{A}}|}\right) \\
&= n \cdot RatioCut(\mathcal{A}) \\
\Rightarrow RatioCut(\mathcal{A}) &= \frac{1}{n} \cdot \mathbf{f}_{\mathcal{A}}^{\top}\mathbf{L}\mathbf{f}_{\mathcal{A}}.
\end{aligned}
$$

(2)

$$
\begin{aligned}
&\mathbf{f}_{\mathcal{A}} \cdot \mathbf{1}_n = |\mathcal{A}|\sqrt{\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|}} - |\bar{\mathcal{A}}|\sqrt{\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|}} = \sqrt{\frac{|\bar{\mathcal{A}}| \cdot |\mathcal{A}|^2}{|\mathcal{A}|}} - \sqrt{\frac{|\mathcal{A}| \cdot |\bar{\mathcal{A}}|^2}{|\bar{\mathcal{A}}|}} = \sqrt{|\bar{\mathcal{A}}| \cdot |\mathcal{A}|} - \sqrt{|\mathcal{A}| \cdot |\bar{\mathcal{A}}|} \\
&\mathbf{f}_{\mathcal{A}} \cdot \mathbf{1}_n = 0 \Leftrightarrow \mathbf{f}_{\mathcal{A}} \perp \mathbf{1}_n.
\end{aligned}
$$

(3)

$$
\|\mathbf{f}_{\mathcal{A}}\|^2 = \sum_{i=1}^{n}(\mathbf{f}_{\mathcal{A}})_i^2 = |\mathcal{A}|\frac{|\bar{\mathcal{A}}|}{|\mathcal{A}|} + |\bar{\mathcal{A}}|\frac{|\mathcal{A}|}{|\bar{\mathcal{A}}|} = |\mathcal{A}| + |\bar{\mathcal{A}}| = n.
$$

B.4 Proof of Proposition 3.3

(The proof is based on [36].)

(1) We have proved that $\mathbf{f}^{\top}\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j} w_{ij}(f_i - f_j)^2$ for any $\mathbf{f}$. Thus we know that $\mathbf{h}_l^{\top}\mathbf{L}\mathbf{h}_l = \frac{1}{2}\sum_{i,j} w_{ij}(h_{i,l} - h_{j,l})^2$. By construction of $\mathbf{h}_l$, if $i, j \in \mathcal{A}_l$ or $i, j \in \bar{\mathcal{A}}_l$, then $h_{i,l} - h_{j,l} = 0$.

Hence,

$$\begin{aligned}
\mathbf{h}_l^\top \mathbf{L}\mathbf{h}_l &= \frac{1}{2}\sum_{i\in\mathcal{A}_l, j\in\bar{\mathcal{A}}_l} w_{ij}(h_{i,l}-0)^2 + \frac{1}{2}\sum_{i\in\bar{\mathcal{A}}_l, j\in\mathcal{A}_l} w_{ij}(0-h_{j,l})^2 \\
&= \frac{1}{2}\sum_{i\in\mathcal{A}_l, j\in\bar{\mathcal{A}}_l} w_{ij}\left(\frac{1}{\sqrt{|\mathcal{A}_l|}}\right)^2 + \frac{1}{2}\sum_{i\in\bar{\mathcal{A}}_l, j\in\mathcal{A}_l} w_{ij}\left(-\frac{1}{\sqrt{|A_l|}}\right)^2 \\
&= \frac{1}{2}\left(\frac{1}{|\mathcal{A}_l|}\right)\cdot cut(\mathcal{A}_l) + \frac{1}{2}\left(\frac{1}{|\mathcal{A}_l|}\right)\cdot cut(\bar{\mathcal{A}}_l) \\
\Rightarrow \mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i &= \frac{cut(\mathcal{A}_i)}{|\mathcal{A}_i|}.
\end{aligned}$$

(2) Obvious.

(3) $RatioCut(\mathcal{A}_1,\ldots,\mathcal{A}_k) = \sum_{i=1}^k \frac{cut(\mathcal{A}_i)}{|\mathcal{A}_i|} = \sum_{i=1}^k \mathbf{h}_i^\top \mathbf{L}\mathbf{h}_i = \sum_{i=1}^k (\mathbf{H}^\top \mathbf{L}\mathbf{G})_{ii} = \mathrm{Tr}(\mathbf{H}^\top \mathbf{L}\mathbf{G})$.

B.5 Proof of Theorem 4.1

(The proof is based on [36].)
Suppose that $\mathbf{A}$ is an $n \times n$ matrix with n linearly independent eigenvectors, and the synchronous affine updates,

$$\mathbf{x}^{(k)} = \mathbf{A}\mathbf{x}^{(k-1)} + \mathbf{b}. \tag{B.1}$$

If $\mathbf{x}^{(k)} \to \mathbf{x}^*$ for $k \to \infty$, then $\mathbf{x}^*$ satisfies $\mathbf{x}^* = \mathbf{A}\mathbf{x}^* + \mathbf{b}$, hence,

$$\mathbf{x}^{(k)} - \mathbf{x}^* = \mathbf{A}\left(\mathbf{x}^{(k-1)} - \mathbf{x}^*\right). \tag{B.2}$$

We define the error $\mathbf{e}^{(k)} \triangleq \mathbf{x}^{(k)} - \mathbf{x}^*$ and we have,

$$\mathbf{e}^{(k)} = \mathbf{A}\mathbf{e}^{(k-1)}. \tag{B.3}$$

Since eigenvectors are linearly independent, they form a basis and thus we can write $\mathbf{e}^{(0)} = \sum_{i=1}^n a_i \mathbf{z}_i$, where $\mathbf{z}_i$ are the eigenvectors with associated eigenvalues λ_i of $\mathbf{A}$. Then

$$\begin{aligned}
\mathbf{e}^{(1)} &= \mathbf{A}\left(\sum_{i=1}^n a_i \mathbf{z}_i\right) = \sum_{i=1}^n a_i \mathbf{A}\mathbf{z}_i = \sum_{i=1}^n a_i \lambda_i \mathbf{z}_i, \\
\mathbf{e}^{(2)} &= \mathbf{A}^2\left(\sum_{i=1}^n a_i \mathbf{z}_i\right) = \mathbf{A}\left(\sum_{i=1}^n a_i \mathbf{A}\mathbf{z}_i\right) = \sum_{i=1}^n a_i \lambda_i \mathbf{A}\mathbf{z}_i = \sum_{i=1}^n a_i \lambda_i^2 \mathbf{z}_i, \\
&\vdots \\
\mathbf{e}^{(k)} &= \mathbf{A}^k\left(\sum_{i=1}^n a_i \mathbf{z}_i\right) = \sum_{i=1}^n a_i \mathbf{A}^k \mathbf{z}_i = \sum_{i=1}^n a_i \lambda_i^k \mathbf{z}_i.
\end{aligned} \tag{B.4}$$

Suppose that $\rho(\mathbf{A}) = |\lambda_n| > |\lambda_i|,\ \forall i = 1, \ldots, n-1$, then

$$\mathbf{e}^{(k)} = a_n \lambda_n^k \mathbf{z}_n + \sum_{i=1}^{n-1} a_i \lambda_i^k \mathbf{z}_i = \lambda_n^k \left(a_n \mathbf{z}_n + \sum_{i=1}^{n-1} a_i \left(\frac{\lambda_i}{\lambda_n} \right)^k \mathbf{z}_i \right). \tag{B.5}$$

Given that $\left| \frac{\lambda_i}{\lambda_n} \right| < 1$, for large k we have that $\left(\frac{\lambda_i}{\lambda_n} \right)^k \simeq 0$, hence

$$\mathbf{e}^{(k)} \simeq \lambda_n^k a_n \mathbf{z}_n. \tag{B.6}$$

Now for the norm of the error, for large k, we have that

$$\|\mathbf{e}^{(k)}\| = \|\lambda_n^k a_n \mathbf{z}_n\| \leq |\lambda_n^k a_n| \|\mathbf{z}_n\|, \tag{B.7}$$

$\mathbf{z}_n$ is an eigenvector of $\mathbf{A}$, so $\|\mathbf{z}_n\| = 1$. Hence,

$$\|\mathbf{e}^{(k)}\| \leq |\lambda_n^k a_n| \leq |\lambda_n|^k |a_n| \Rightarrow \|\mathbf{e}^{(k)}\| \leq \rho(\mathbf{A})^k |a_n|. \tag{B.8}$$

Thus, for smaller spectral radius, $\rho(\mathbf{A})$, we have faster convergence.

B.6 Proof of Theorem 4.2

Suppose that $\mathbf{A}$ is an $n \times n$ matrix with n linearly independent eigenvectors, and the asynchronous affine updates,

$$\begin{aligned} \mathbf{x}^{(k)} &= \mathbf{\Psi}^{(k)} \left(\mathbf{A}\mathbf{x}^{(k-1)} + \mathbf{b} \right) + (\mathbf{I} - \mathbf{\Psi}^{(l)}) \mathbf{x}^{(k-1)} \\ &= \left(\mathbf{\Psi}^{(k)} \mathbf{A} + \mathbf{I} - \mathbf{\Psi}^{(k)} \right) \mathbf{x}^{(k-1)} + \mathbf{\Psi}^{(k)} \mathbf{b} \\ &= \mathbf{B}^{(k)} \mathbf{x}^{(k-1)} + \mathbf{\Psi}^{(k)} \mathbf{b}, \text{ for } \mathbf{B}^{(k)} = \mathbf{\Psi}^{(k)} \mathbf{A} + \mathbf{I} - \mathbf{\Psi}^{(k)}. \end{aligned} \tag{B.9}$$

If $\mathbf{x}^{(k)} \to \mathbf{x}^*$ for $k \to \infty$, then $\mathbf{x}^*$ satisfies $\mathbf{x}^* = \mathbf{B}^{(k)} \mathbf{x}^* + \mathbf{\Psi}^{(k)} \mathbf{b}$, hence,

$$\mathbf{x}^{(k)} - \mathbf{x}^* = \mathbf{B}^{(k)} \left(\mathbf{x}^{(k-1)} - \mathbf{x}^* \right) \tag{B.10}$$

We define the error $\mathbf{e}^{(k)} \triangleq \mathbf{x}^{(k)} - \mathbf{x}^*$ and we have,

$$\begin{aligned} \mathbf{e}^{(k)} &= \mathbf{B}^{(k)} \mathbf{e}^{(k-1)} \\ \Rightarrow \mathbb{E}[\mathbf{e}^{(k)}] &= \mathbb{E}[\mathbf{B}^{(k)} \mathbf{e}^{(k-1)}] = \mathbb{E}[\mathbf{B}^{(k)}] \mathbb{E}[\mathbf{e}^{(k-1)}], \quad (\mathbf{B}^{(k)}, \mathbf{e}^{(k-1)} \text{ independent}) \\ \Rightarrow \mathbb{E}[\mathbf{e}^{(k)}] &= (\mathbf{P}(\mathbf{A} - \mathbf{I}) + \mathbf{I}) \cdot \mathbb{E}[\mathbf{e}^{(k-1)}], \quad (\mathbb{E}[\mathbf{B}^{(k)}] = \mathbf{P}(\mathbf{A} - \mathbf{I}) + \mathbf{I}) \\ \Rightarrow \mathbb{E}[\mathbf{e}^{(k)}] &= \bar{\mathbf{A}} \cdot \mathbb{E}[\mathbf{e}^{(k-1)}], \quad (\bar{\mathbf{A}} = \mathbf{P}(\mathbf{A} - \mathbf{I}) + \mathbf{I})) \end{aligned} \tag{B.11}$$

We can write $\mathbf{e}^{(0)} = \sum_{i=1}^n a_i \mathbf{z}_i$, where $\mathbf{z}_i$ are the eigenvectors with associated eigenvalues λ_i of $\bar{\mathbf{A}}$. Then

$$\mathbb{E}[\mathbf{e}^{(1)}] = \bar{\mathbf{A}} \cdot \mathbb{E}[\mathbf{e}^{(0)}] = \bar{\mathbf{A}} \cdot \mathbb{E}\left[\sum_{i=1}^n a_i \mathbf{z}_i \right] = \mathbb{E}\left[\sum_{i=1}^n a_i \bar{\mathbf{A}} \mathbf{z}_i \right] = \mathbb{E}\left[\sum_{i=1}^n a_i \lambda_i \mathbf{z}_i \right],$$

$$\mathbb{E}[\mathbf{e}^{(2)}] = \bar{\mathbf{A}}^2 \cdot \mathbb{E}[\mathbf{e}^{(0)}] = \bar{\mathbf{A}}^2 \cdot \mathbb{E}\left[\sum_{i=1}^{n} a_i \mathbf{z}_i\right] = \mathbb{E}\left[\sum_{i=1}^{n} a_i \bar{\mathbf{A}}^2 \mathbf{z}_i\right] = \mathbb{E}\left[\sum_{i=1}^{n} a_i \lambda_i^2 \mathbf{z}_i\right],$$

$$\vdots$$

$$\mathbb{E}[\mathbf{e}^{(k)}] = \bar{\mathbf{A}}^k \cdot \mathbb{E}[\mathbf{e}^{(0)}] = \bar{\mathbf{A}}^k \cdot \mathbb{E}\left[\sum_{i=1}^{n} a_i \mathbf{z}_i\right] = \mathbb{E}\left[\sum_{i=1}^{n} a_i \bar{\mathbf{A}}^k \mathbf{z}_i\right] = \mathbb{E}\left[\sum_{i=1}^{n} a_i \lambda_i^k \mathbf{z}_i\right]. \tag{B.12}$$

Suppose that $\rho(\bar{\mathbf{A}}) = |\lambda_n| > |\lambda_i|,\ \forall i = 1, \ldots, n-1$, then

$$\mathbb{E}[\mathbf{e}^{(k)}] = \mathbb{E}\left[a_n \lambda_n^k \mathbf{z}_n + \sum_{i=1}^{n-1} a_i \lambda_i^k \mathbf{z}_i\right] = \mathbb{E}\left[\lambda_n^k \left(a_n \mathbf{z}_n + \sum_{i=1}^{n-1} a_i \left(\frac{\lambda_i}{\lambda_n}\right)^k \mathbf{z}_i\right)\right]. \tag{B.13}$$

Given that $\left|\frac{\lambda_i}{\lambda_n}\right| < 1$, for large k we have that $\left(\frac{\lambda_i}{\lambda_n}\right)^k \simeq 0$, hence

$$\mathbb{E}[\mathbf{e}^{(k)}] \simeq \mathbb{E}\left[\lambda_n^k a_n \mathbf{z}_n\right] = \lambda_n^k a_n \mathbf{z}_n. \tag{B.14}$$

Now for the norm of the error, for large k, we have that

$$\left\|\mathbb{E}\left[\mathbf{e}^{(k)}\right]\right\| = \|\lambda_n^k a_n \mathbf{z}_n\| \leq |\lambda_n^k a_n| \|\mathbf{z}_n\|, \tag{B.15}$$

$\mathbf{z}_n$ is an eigenvector of $\bar{\mathbf{A}}$, so $\|\mathbf{z}_n\| = 1$. Hence,

$$\left\|\mathbb{E}\left[\mathbf{e}^{(k)}\right]\right\| \leq |\lambda_n^k a_n| \leq |\lambda_n|^k |a_n| \Rightarrow \left\|\mathbb{E}\left[\mathbf{e}^{(k)}\right]\right\| \leq \rho(\bar{\mathbf{A}})^k |a_n|. \tag{B.16}$$

Thus, for smaller spectral radius, $\rho(\bar{\mathbf{A}})$, we have faster convergence.

Bibliography

[1] J. S. Yedidia, "Message-passing algorithms for inference and optimization," *Journal of Statistical Physics*, vol. 145, no. 4, pp. 860–890, 2011.

[2] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[3] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[4] G. Vannucci, A. Bletsas, and D. Leigh, "A software-defined radio system for backscatter sensor networks," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2170–2179, Jun. 2008.

[5] C. Konstantopoulos, E. Kampianakis, E. Koutroulis, and A. Bletsas, "Wireless sensor node for backscattering electrical signals generated by plants," in *Proc. IEEE Sensors Conf. (Sensors)*, Baltimore, MD, USA, Nov. 2013.

[6] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," in *Proc. ACM SIGCOMM*, Hong Kong, China, 2013, pp. 39–50.

[7] J. Kimionis, A. Bletsas, and J. N. Sahalos, "Increased range bistatic scatter radio," *IEEE Trans. Commun.*, vol. 62, no. 3, pp. 1091–1104, Mar. 2014.

[8] E. Kampianakis, J. Kimionis, K. Tountas, C. Konstantopoulos, E. Koutroulis, and A. Bletsas, "Wireless environmental sensor networking with analog scatter radio & timer principles," *IEEE Sensors J.*, vol. 14, no. 10, pp. 3365—3376, Oct. 2014.

[9] S. N. Daskalakis, S. D. Assimonis, E. Kampianakis, and A. Bletsas, "Soil moisture wireless sensing with analog scatter radio, low power, ultra-low cost and extended communication ranges," in *Proc. IEEE Sensors Conf. (Sensors)*, Valencia, Spain, Nov. 2014, pp. 122–125.

[10] N. Fasarakis-Hilliard, P. N. Alevizos, and A. Bletsas, "Coherent detection and channel coding for bistatic scatter radio sensor networking," in *Proc. IEEE Int. Conf. Communications*, Jun. 2015, pp. 4895–4900.

[11] S. N. Daskalakis, S. D. Assimonis, E. Kampianakis, and A. Bletsas, "Soil moisture scatter radio networking with low power," *IEEE Trans. Microwave Theory Tech.*, vol. 64, no. 7, pp. 2338–2346, Jul. 2016.

[12] V. Papageorgiou, A. Nichoritis, P. Vasilakopoulos, G. Vougioukas, and A. Bletsas, "Towards ambiently powered internet-of-things-that-think with asynchronous principles," School of Electrical and Computer Engineering, Technical University of Crete, Tech. Rep., 2022.

[13] ——, "Towards ambiently powered inference on wireless sensor networks: Asynchrony is the key!" in *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2021, pp. 458–465.

[14] D. Bickson, "Gaussian belief propagation: Theory and aplication," *arXiv preprint arXiv:0811.2518*, 2008.

[15] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[16] D. Shah, "Algorithms for inference—mit course no. 6.438," Cambridge MA, 2014, MIT OpenCourseWare. [Online]. Available: https://ocw.mit.edu/courses/6-438-algorithms-for-inference-fall-2014/

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[18] B. J. Frey, F. R. Kschischang, H.-A. Loeliger, and N. Wiberg, "Factor graphs and algorithms," in *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, vol. 35. Citeseer, 1997, pp. 666–680.

[19] B. Li and Y.-C. Wu, "Convergence analysis of gaussian belief propagation under high-order factorization and asynchronous scheduling," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 2884–2897, 2019.

[20] C. D. Meyer, *Matrix analysis and applied linear algebra*. Siam, 2000, vol. 71.

[21] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. USA: Prentice-Hall, Inc., 1989.

[22] O. Teke and P. P. Vaidyanathan, "Random node-asynchronous graph computations: Novel opportunities for discrete-time state-space recursions," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 64–73, 2020.

[23] ——, "Randomized asynchronous recursions with a sinusoidal input," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 1491–1495.

[24] ——, "Random node-asynchronous updates on graphs," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 2794–2809, 2019.

[25] J. MacQueen, "Classification and analysis of multivariate observations," in *5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[26] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[27] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[28] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 14, 2001.

[29] L. Helmut, *Handbook of Matrices*. Wiley, 1997.

[30] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[31] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[32] A. Y. Kibangou and C. Commault, "Decentralized laplacian eigenvalues estimation and collaborative network topology identification," *IFAC Proceedings Volumes*, vol. 45, no. 26, pp. 7–12, 2012.

[33] T.-M.-D. Tran and A. Y. Kibangou, "Distributed estimation of laplacian eigenvalues via constrained consensus optimization problems," *Systems & Control Letters*, vol. 80, pp. 56–62, 2015.

[34] M. Yang and C. Y. Tang, "Distributed estimation of graph spectrum," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 2703–2708.

[35] A. Nichoritis, "Design and implementation of a solar powered wireless sensor network for autonomous inference," Diploma Thesis, Technical University of Crete, 2021.

[36] W. Ford, *Numerical linear algebra with applications: Using MATLAB*. Academic Press, 2014.