DIPLOMARBEIT

CONSENSUS ALGORITHMS FOR NETWORKED CONTROL

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplomingenieurs

unter der Leitung von

Ao. Univ.Prof. Dipl.-Ing. Dr. Gerald Matz

Univ.Ass. Dipl.-Ing. Dr. Valentin Schwarz

Dipl.-Ing. Gábor Hannák

Institut für Nachrichtentechnik und Hochfrequenztechnik

eingereicht an der Technischen Universität Wien

Fakultät für Elektrotechnik und Informationstechnik

von

Jorge Almela Miralles

Wien, Juli 2014

Acknowledgement

Dedicado a mi familia,

por su apoyo a mi aventura lejos de casa.

Pre Janu,

za jej bezpodmienečnú podporu a za to,

že mi celý čas verila.

To Gerald, Valentin and Gábor,

for giving me the opportunity of working

and completing my thesis with them.

Abstract

This thesis deals with distributed consensus algorithms for multi-agent networks. The goal of consensus algorithms is to achieve a consensus for a specific task. In our case we consider a setting where network agents should meet at a single point. This rendezvous algorithm is well known in literature. We mainly focus on two dimensional random geometric networks, but one dimensional networks and regular scenarios are also considered. For the performance study we pay special attention to the convergence properties and the ability of the agents to reach the consensus point. All tasks carried out by the nodes are distributed, in other words, there is no need of external agents for coordinating or deciding. In addition to the consensus algorithms the effect of using control within these networks is also studied. With these control contributions we try to improve the overall performance of the system. We will try to control the dynamics of the nodes and also apply temporal delays to the movement. Finally, we present some numerical results where we can observe how different settings and parameters affect the behavior of the whole system.

Keywords: Multi-agent wireless networks, consensus algorithms, constant weights, Metropolis-Hastings weights, random geometric graphs, control formation, MSE.

Zusammenfassung

Diese Diplomarbeit befasst dich mit verteilten Konsensalgorithmen in Netzwerken mit mehreren Agenten. Das Ziel dieser Algorithmen ist das Erreichen eines Konsenses für eine gegebene Aufgabe. In dieser Arbeit beschäftigen wir uns mit Netzwerken, dessen bewegliche Agenten sich in einem Punkt des Raumes treffen wollen. Algorithmen, welche genau diese Aufgabe meistern, werden Rendezvous-Algorithmen genannt und gehören zu den Konsensalgorithmen. Unser Fokus liegt auf zweidimensionalen geometrischen Graphen mit zufälliger Knotenverteilung, wobei auch eindimensionale und reguläre Strukturen untersucht werden. Bei der Auswertung der Güte dieser Algorithmen konzentrieren wir auf die Konvergenzeigenschaften und auf die Erfolgsrate, dass sich alle Agenten in genau einem Punkt treffen. Im Netzwerk ist keine zentrale Recheneinheit vorhanden, somit werden alle Schritte der Algorithmen verteilt von den Agenten ausgeführt. Des weiteren untersuchen wir Kontrollmethoden wie Lenkung und zeitliche Verzögerung, welche sicherstellen sollen, dass sich alle Agenten in einem Punkt treffen. Zum Schluss wird der Einfluss der Parameter auf die eingeführten Kontroll- und Konsensalgorithmen durch diverse Simulationen anschaulich gemacht.

Abstract

Spanish version

En este proyecto contemplamos el desarrollo de nuevos algoritmos de consenso para redes inalámbricas con múltiples agentes, así como, el planteamiento de modificaciones sobre los mismos con el fin de optimizarlos. El objetivo de los algoritmos de consenso es conseguir cooperativamente un acuerdo para una determinada tarea. Nos centramos en redes de dos dimensiones con distribuciones aleatorias de sus nodos, también analizamos redes unidimensionales y tratamos algunos casos de redes con distribución regular. Hemos estudiado algoritmos de consenso extraídos de otros trabajos científicos, cuyo rendimiento hemos comparado con otros desarrollados por nosotros mismos. Hemos llevado un seguimiento especial a las propiedades de convergencia y a la capacidad de los agentes de reunirse en un punto. Todas las tareas que realizan los agentes son distribuidas, es decir, no hay ninguna necesidad de que un agente externo intervenga para coordinar o decidir los movimientos. Además de los algoritmos de consenso, también hemos analizado el efecto de aplicar control para tratar de mejorar el comportamiento de las redes. Este proceso de control está justificado por la necesidad de mejorar la efectividad de los algoritmos de consenso y está dirigido a controlar el movimiento de los nodos y a la aplicacón de ciertos retardos al mismo. Tras estudiar cuidadosamente los algoritmos de consenso y las modificaciones realizadas sobre ellos, pasamos a mostrar algunos de los resultados numéricos, gráficas y figuras donde se puede observar como la elección de diferentes inicializaciones o parámetros de entrada afectan al comportamiento general del sistema.

Contents

List of Abbreviations

- **CW:** Constant weights
- **MH:** Metropolis Hastings weights
- **RV:** Rendezvous weights
- **AW:** Angular weights
- **MSE:** Mean-squared error
- **W:** Weight matrix
- **1D:** One dimensional scenario
- **2D:** Two dimensional scenario

1 Introduction

The purpose of this thesis is the design and analysis of consensus algorithms for multi-agent systems and the creation of control methods for improving the connectivity and convergence results of them. By multi-agent systems we understand groups of nodes forming networks. These agents have wireless connectivity which let them exchange information and besides, they have some computational capabilities and power supply units so they could accomplish tasks with a certain autonomy. We got inspired by the works [1–4], they help us to start with the work and to set several goals.

We will present a framework where the algorithms are tested and also the theory background for understanding how they work. The programming environment chosen is Matlab both for coding and for testing.

With regard to consensus algorithms, we test in this work three different methods, taken from other scientific works, which are constant weights [3], Metropolis-Hastings [5] and rendezvous [4], for then creating new methods which try to improve them and correct their flaws. These new methods will be called angular weights (for 2D scenarios) and linear weights (for 1D scenarios).

The control contributions of the thesis are applied over the consensus algorithms in order to improve their behavior. Adding delay, control the dynamics or creating favorable network distributions are some of the ideas. The control improvements will be tested over the rendezvous algorithm.

The consensus algorithms are distributed applications which seek a cooperative agreement for solving a problem. This thesis will mainly focus on the rendezvous problem, or in other words, decide a meeting point depending on the states of the agents of the network under study. This type of algorithms is based on graph and matrix concepts which will be further presented.

The work is composed of different parts. First a study of the consensus algorithms with different weight matrices will be done. This study will help to understand and know better the behavior of the algorithm depending on the different parameters.

The second part will be creating new weights matrices for the consensus algorithms looking for a better performance than the old. Then the results of these new methods compared with the old weight matrix will be shown. The last step of the work will be the application of control modifications in order to improve the features of the consensus algorithm. Finally the results will show whether the control of the network improves or not the overall performance.

The thesis is structured as follows:

Chapter 2: This chapter gives a basic introduction to graph theory. Moreover, some aspects about linear consensus and convergence will be discussed.

Chapter 3: This part of the work deals with the problem statement and the consensus algorithm.

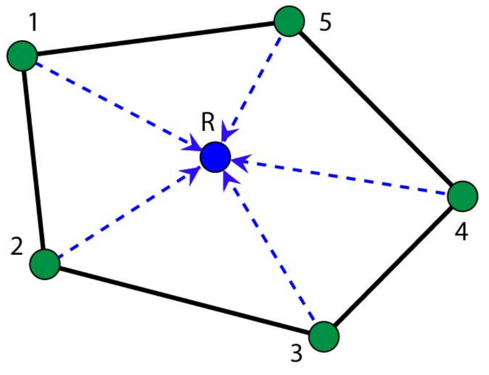
Chapter 4: In this chapter some modifications of the Rendezvous algorithm are presented.

Chapter 5: The results and graphics appear in this chapter.

Chapter 6: In the last part, conclusions are provided and an outlook for further research is given.

1.1 Motivating Example

As a piece of example, before starting with the theory we will have a look at Fig. 1.1 where what is called Rendezvous in this thesis is described graphically. In Fig. 1.1



Blue arrows: trajectories
Black lines: communication links
1 . . . 5: nodes
R: agreed meeting point

Figure 1.1: *Rendezvous*

are shown some agents or nodes which can communicate and share state information. The goal is with this flow of data and without an external agent achieve a cooperative agreement on where to meet. This agreed position will be called either rendezvous or consensus point and in the case of the current example is the blue point in the middle. The nodes will follow different trajectories depending on the weight matrix design used. The analysis of the weights matrices and their advantages and drawbacks will be therefore another focus of attention in this work.

2 Prerequisites

This chapter will be used for giving an approach to the different required theoretical concepts. We will give an insight to graph theory, graph topology and matrix theory. The different parts of the introduction can be complemented for the interested reader with the works [6–8] dealing graph theory, the articles [9–12] for matrix theory and regarding graph topology we suggest [13]. The following chapters of the work will stand over these basis so we will include some examples for making them easier to understand.

2.1 Graph Theory

In a communication network, it is not necessary that all the agents have direct communication. It is more important if there always exist a communication link between each pair of nodes of the net. For knowing how a network behaves or copes with possible issues as for instance when a link fails, the mathematical concept of the graph can be used.

A graph is a set of nodes, and a set of edges which represents the connectivity of the network and the number of possible ways to reach a given agent from any other.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \tag{2.1}$$

In the equation (2.1) we see how $\mathcal{G}$ is defined containing a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. The set of nodes $\mathcal{V} = \{1, \ldots, I\}$ contains the elements of the graph, whereas the set of edges $\mathcal{E}$ denote which nodes have links between them.

2.1.1 Definitions

- Undirected graph

  For simplicity it is assumed that all the nodes can transmit the same power and thus have the same scope. This permits to use only undirected graphs for describing the behavior of the networks. Hereafter in this thesis all the graphs refer to undirected graphs.

  An undirected graph has nodes connected by undirected edges. That means that the edges have no tail nor head, therefore they have no direction and if i can reach j they can obtain information from each other. In other words:

$$e_{ij} = e_{ji} \tag{2.2}$$

It is possible to see in Fig. 2.1 an example of undirected graph with $\mathcal{V} = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{E} = \{(1, 2), (1, 3), (2, 4), (2, 5), (3, 4), (3, 6), (4, 5), (4, 6)\}$. For instance, in this case the node 1 is linked with 2 and 3, therefore from 2 and 3 is possible to establish a two-way communication with node 1 and condition (2.2) is fulfilled. The

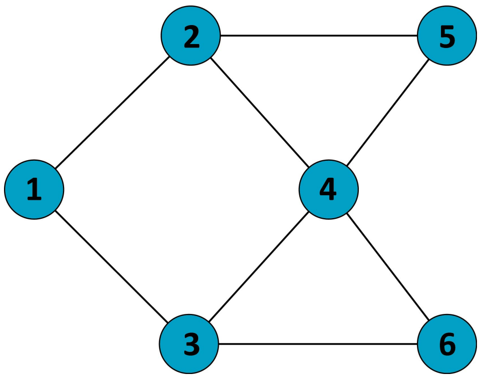


Figure 2.1: *Undirected graph*

same graph will be used along the whole Section 2.1.

- Path

  A path is a finite or infinite sequence of edges connecting two nodes. Continuing with the example, in Fig. 2.2 it is shown a path joining the node 1 with the node 6. This path contains the edges $e_{1,3}$ and $e_{3,6}$. Since we are working with undirected

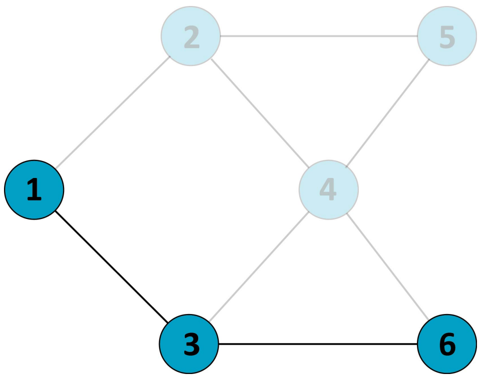


Figure 2.2: *Path*

graphs, it is easy to notice that, as condition 2.2 implies, having a path from node 1 to node 6 means that the same path exists from node 6 to node 1.

- Neighbor set and degree

  The neighbor set of a given node i, $\mathcal{N}(i)$, contains agents within its range. The neighbor set for an undirected graph can be defined as

$$\mathcal{N}(i) = \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\} \tag{2.3}$$

  where we see that (2.3) means that the node j belongs to the neighbor set of i if and only if there is an edge between them.

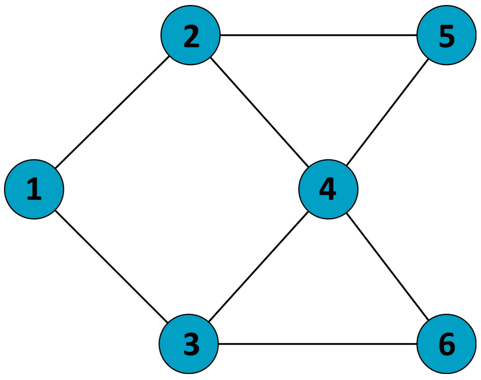
  The number of elements in the neighbor set of i is the degree of node i $d(i)$, i.e., $d(i) = |\mathcal{N}(i)|$.

  In our example Fig. 2.1 the neighbor set of node 1 is $\mathcal{N}(1) = \{2, 3\}$ and its degree $d(1) = 2$.

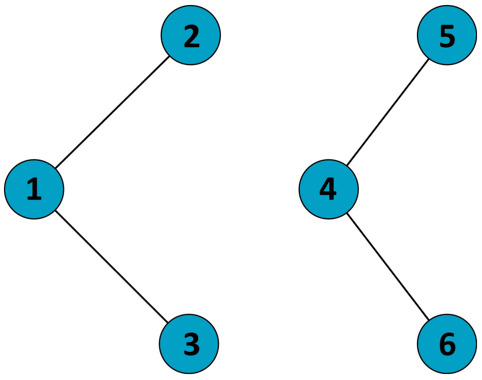
- Connectivity

  In graph theory it is said that two nodes are connected if there is at least a path joining them. Therefore a graph $\mathcal{G}$ is connected if there is a path from each node to all others. In any other case $\mathcal{G}$ would be disconnected. Furthermore, for the case of undirected graphs we can say that if there is a path between each pair of nodes then the graph is always strongly connected.

  Now in Fig. 2.3 there is a comparison between a connected and an unconnected graph. We can appreciate how in the graph a it is possible to reach from any node any other, whereas with the graph b this is not possible, i.e., from 1 it is not feasible to reach 6.



(a) *Connected*



(b) *Unconnected*

Figure 2.3: *Comparing graphs*

2.1.2 Adjacency Matrix

The adjacency matrix $\mathbf{A}$ is a way to define a graph $\mathcal{G}$ by setting the entries of the matrix which correspond to connected edges to one and the others to zero. The adjacency matrix $\mathbf{A}$ has $I \times I$ entries where $I = |\mathcal{V}|$ is the number of nodes of the graph. The equation (2.4) describes the construction of $\mathbf{A}$.

$$[\mathbf{A}]_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{else} \end{cases} \tag{2.4}$$

Since the edges are bidirectional, for undirected graphs the adjacency matrix $\mathbf{A}$ is always symmetric.

For a better understanding, let us define the adjacency matrix $\mathbf{A}$ of the network of Fig. 2.4. This graph will help us along Sections 2.1.2 to 2.1.4, and we will support the theoretical explanation with practical results. Using Fig. 2.4 is easy to define the corresponding adjacency matrix $\mathbf{A}$. For node 1 there are two links with node 2 and node 3. This leads to set the entries of the first row of $\mathbf{A}$ as follows:

$$a_{12} = a_{13} = 1$$

$$a_{14} = a_{15} = a_{16} = a_{17} = a_{18} = 0$$

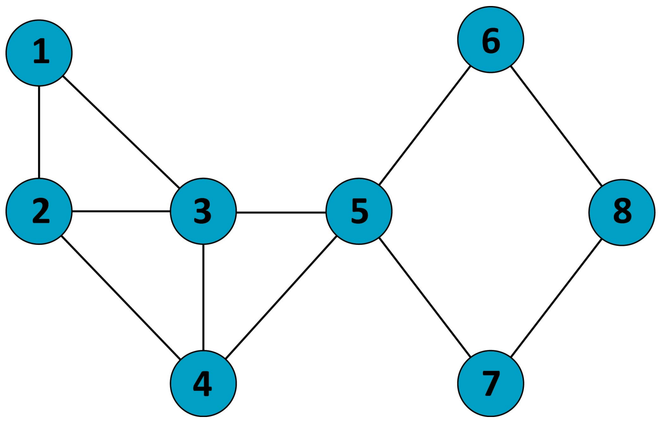


Figure 2.4: *Undirected graph*

There are not self loops, thus $a_{ii} = 0$ for all i. Eventually, the adjacency matrix $\mathbf{A}$ is:

$$[\mathbf{A}]_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

By simple visual check it is possible to notice that the matrix is symmetric as expected.

2.1.3 Degree Matrix

The degree matrix $\mathbf{D}$ of a graph $\mathcal{G}$ is a diagonal matrix containing the number of neighbors. Again, the dimension is $I \times I$. The entries of the diagonal of this matrix can be calculated with the sum of the elements of the rows or columns of the adjacency matrix and the rest of the elements equal to 0, then the matrix is constructed as in (2.5).

$$[\mathbf{D}]_{ij} = \begin{cases} \mathcal{N}(i) & \text{if } i = j \\ 0 & \text{else} \end{cases} \tag{2.5}$$

and consequently the degree matrix takes the following structure,

$$[\mathbf{D}]_{ij} = \begin{pmatrix} d(1) & 0 & \cdots & 0 \\ 0 & d(2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & d(I) \end{pmatrix}$$

As a piece of example, taking Fig. 2.4 it is possible to obtain the matrix below.

$$[\mathbf{D}]_{ij} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

If we now count the number of neighbors of each node we will see that they correspond to its value on the diagonal, i.e.,

$$\mathcal{N}(1) = \mathbf{D}_{11}.$$

2.1.4 Laplacian Matrix

The last possible representation of a graph $\mathcal{G}$ which is going to be presented is the Laplacian matrix $\mathbf{L}$, which is a $I \times I$ matrix.

For an undirected graph the entries of the Laplacian matrix $\mathbf{L}$ are given by

$$[\mathbf{L}]_{ij} = \begin{cases} \mathrm{d}(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \quad \text{and } (i,j) \in \mathcal{E} \\ 0 & \text{else} \end{cases}$$

It can be also represented as the difference of the degree matrix $\mathbf{D}$ and the adjacency matrix $\mathbf{A}$:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.6}$$

Since the the degree matrix $\mathbf{D}$ is diagonal and the adjacency matrix $\mathbf{A}$ is symmetric the result will be a symmetric Laplacian matrix $\mathbf{L}$.

There are some spectral properties of the Laplacian matrix $\mathbf{L}$ which shall be mentioned:

- The sum of each row of the Laplacian matrix is zero, therefore the right eigenvector of $\mathbf{L}$ is $\mathbf{1}$.

$$\mathbf{L}\mathbf{1} = 0.$$

- For undirected graphs, the Laplacian matrix is positive semi-definite, i.e., for all $\mathbf{x} \in \mathcal{R}^I$,

$$\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0.$$

- According to Gershgorin theorem all eigenvalues of $\mathbf{L}$ in the complex plane are located in a closed disk centered at $\Delta + 0j$ with a radius of $\Delta = d_{\max}(i)$, i.e. the maximum degree of the graph.

- For undirected graphs the Laplacian matrix is symmetric and has I real eigenvalues which can be ordered in ascending order as

$$0 = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_i \leq 2\Delta.$$

The zero eigenvalue is known as the trivial eigenvalue of $\mathbf{L}$. The second eigenvalue λ_2 needs to be greater than zero for a connected graph. It is known as the algebraic connectivity of the network, moreover, it is a measure of performance/speed of consensus algorithms. It is possible to know the number of components of the graph which are disconnected because it coincides with the number of 0 eigenvalues.

For getting a better understanding of the concepts presented before, it is time to use Fig. 2.4 and give a practical example. Using (2.6) the resulting matrix $\mathbf{L}$ is:

$$[\mathbf{L}]_{ij} = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{pmatrix}$$

Finally, the eigenvalues are obtained

$$\lambda_i = \{0, 0.5, 2, 2.1, 3, 4, 4.8, 5.6\} \quad \text{for} \quad i \in [1, 8],$$

there is only one zero eigenvalue, hence can be concluded that the graph is connected. The algebraic connectivity in this case is $\lambda_2 = 0.5$.

2.2 Network Topology

In this Section the random geometric topology will be described following some of the ideas given in [14], because it is the topology used for the simulations. In Fig. 2.5 it is possible to see a random geometric graph. A random geometric graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$,

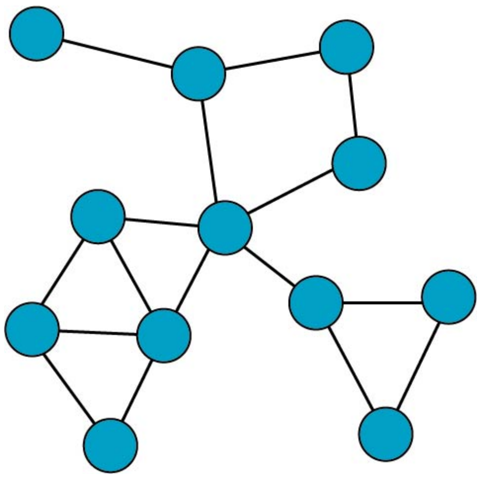


Figure 2.5: *Random geometric graph*

where $I = |\mathcal{V}|$ is the number of nodes which are randomly distributed, in our case, over the unit square. The maximum length of the edges will be the connectivity radius r or the range, that means that the maximum Euclidean distance between two connected nodes i and j will be $d(i,j) = r$. Then the adjacency matrix of a random geometric graph is as below:

$$a_{ij} = a_{ji} = \begin{cases} 1 & \text{if} \quad d(i,j) \leq r, \\ 0 & \text{else.} \end{cases}$$

2.2.1 Other Network Topologies

For some cases, special network distributions will be needed. Concretely, in this thesis we will use a linear distribution for analyzing the behavior of the consensus algorithms in one dimensional scenarios and also regular grid topologies to study possible improvements on the performance when the nodes are evenly distributed.

3

Consensus Algorithms

In this work one of the main goals is the analysis of consensus algorithms for multi-agent networked systems, and concretely, their application in Rendezvous problems and control formation. It is possible to learn more about consensus algorithms for multi-agent networks in [15].

In networks of agents, consensus means to cooperatively reach an agreement taking into account diverse interests depending on the state of the agents. Hence, we will use distributed computation. As a consequence, we can achieve benefits such as higher efficiency or better computational capabilities. The distributed algorithms are widely used in applications like surveillance, spaced-based interferometers, reconnaissance systems or distributed re-configurable sensor networks.

3.1 Problem Statement

In this section we will explain the framework of study and the objectives. Let us assume that there is a network with I randomly distributed wireless agents equipped with memory and computing capabilities. These agents have the same communication range and know the current state of their neighbors. Therefore the network can be studied as an undirected graph of I nodes with a random geometric initial state. Moreover, we will assume the graph to be always initially connected.

Once the scenario is initialized the nodes must cooperatively reach consensus using a consensus algorithm and a weight matrix $\mathbf{W}$ among one of the designs we will present in Section 3.5. Therefore the nodes will move, and their new positions will be a linear combination of the positions of the neighbors and themselves.

In most of the cases the objective for us will be achieving Rendezvous, so the aim with which the weight matrix $\mathbf{W}$ will be designed is making the nodes to converge in a single point. However, we will use control formation as a case of study for improving the performance (further explanation in Section 4) and then the goal for the weight matrix $\mathbf{W}$ will be making the nodes keep a distance d between them. We recall that the only available information for them is their current state and the current state of their neighbors. Hence, the consensus algorithm is launched as a distributed application because all the nodes decide where to move based on the information they have.

Finally, when the objectives and the scenario are explained, it is important to know that the main problems we are coping with are the following:

- The movement of the nodes can sometimes lead to the disconnection of the network, having as a result local consensus instead of a unique Rendezvous point.
- Some of the weight designs make the nodes movement too slow making unfeasible

to converge.

Additionally, in Section 4 some control features will be implemented in order to improve the performance of the algorithm, and face the mentioned problems. In Section 5 we will show the actual performance of these modifications and whether they help or not.

At the end we will analyze what happens if the network is regular and not random geometric, concretely hexagonal and squared grids and, despite we will focus in two dimensional scenarios, also linear distributions will be studied. For the case of one dimensional networks we will develop a special algorithm which will be run over regular and random scenarios.

3.2 Consensus in Discrete -Time and Matrix Theory

Here we will present a framework for analysis of convergence of consensus algorithm for networks with fixed topology in discrete time following the guidelines stated in [3].

As said we will focus on discrete time consensus algorithm, where k denotes each iteration. Let us define the inputs of the algorithm with $\mathbf{r}_i$ the position of the agent i at time 0, where $\mathbf{x}[0]$ is the initial inner state vector containing all the agents positions at time 0:

$$\mathbf{x}_i[0] = \mathbf{r}_i.$$

Let also the weight matrix $\mathbf{W}[k] \in \mathbb{R}^{I \times I}$ be a square matrix with entries w_{ij}. Then the consensus algorithm is as

$$\mathbf{x}_i[k] = w_{ii}[k]\mathbf{x}_i[k-1] + \sum_{j \in \mathcal{N}_i} w_{ij}[k]\mathbf{x}_j[k-1], \quad i = 1, \ldots, I \tag{3.1}$$

Accordingly to (3.1), the information state of each agent is updated as the weighted sum of its current state and the current states of its neighbors. Thus, there is only communication between first-order neighbors. Note that if the agent is isolated its state information will remain in the next time step.

Now the discrete time collective dynamics of the network under this algorithm can be stated as:

$$\mathbf{x}[k] = \mathbf{W}[k]\mathbf{x}[k-1]. \tag{3.2}$$

At this point, and before presenting some important properties of the weight matrix **W** we will present its construction. For doing so we will use the simplest design for **W**, this is the constant weights approach. We will give a wider and more detailed explanation in Section 3.5.3.

$$\mathbf{W} = \mathbf{L} - \alpha\mathbf{I} \tag{3.3}$$

In the equation (3.3) we see the Laplacian matrix **L**, presented in Section 2.1.4, the identity matrix **I** and finally a new term α which we will call step-size. The value α will play and important role in the convergence properties as we will explain afterwards in this section. The step-size needs to be whithin certain boundaries for achieving convergence, in this section we will work with the boundaries for CW which are $\alpha \in (0, \frac{1}{\Delta}]$.

We will define some of the properties which the matrix **W** should have. It must be non-negative, in other words a matrix where all its entries are positive or equal to 0. Using the Perron-Frobenius theorem we will discuss some properties of the non-negative matrices. There are three important types of non-negative matrices, such are irreducible, stochastic and primitive.

- A matrix **W** is irreducible if its associated graph is connected.
- A non-negative matrix is called row or column stochastic if the sum of its rows or

columns equals 1.

- Finally an irreducible stochastic matrix is primitive if it has only one eigenvalue with maximum modulus.

Then the $\mathbf{W}$ of the graph $\mathcal{G}$ with I nodes, maximum degree Δ and $\alpha \in (0, \frac{1}{\Delta}]$ satisfies the following properties.

- It is row stochastic non-negative with a trivial eigenvalue 1.
- All its eigenvalues are in the complex unit circle. Note that eigenvalues of $\mathbf{W}$ and $\mathbf{L}$ are the same where

  $$\lambda_{\mathbf{W}} = 1 - \alpha\lambda_{\mathbf{L}},$$

  and using again Gershgorin theorem, all eigenvalues of $\mathbf{L}$ are in a closed disk $|s - \Delta| < \Delta$, therefore defining $z = 1 - \frac{s}{\Delta}$ we have radius $|z| \leq 1$ or equivalently, the eigenvalues of $\mathbf{W}$ inside a unit circle.
- If $\mathcal{G}$ is connected and $0 < \alpha < \frac{1}{\Delta}$, then $\mathbf{W}$ is a primitive matrix.

The condition $\alpha < \frac{1}{\Delta}$ is necessary because a larger step-size would lead into having a non-primitive $\mathbf{W}$ matrix.

3.3 Algorithm

Over the scenario described in Section 3.1 and under the conditions explained in Section 3.2, the consensus algorithm works in some well differentiated steps.

(i) First, as explained before the nodes are randomly distributed.

(ii) Then the nodes search for their neighbors, this information is for us the adjacency matrix since it contains the connectivity information.

(iii) When all the information is gathered, it is time to compute the weights w_{ij}. They will eventually determine where the nodes will move.

(iv) At this step it is already possible to implement the update (3.1) with which it is possible to compute the new position.

(v) Then it is time for the nodes to move to their new locations.

These steps will repeat until we reach a predefined number of iterations or consensus is achieved.

3.4 Convergence Conditions

The consensus equation described in (3.1) can also be written in a matrix formulation, and this way we can explain the convergence conditions which $\mathbf{W}$ must fulfill for the general case with I dimensions,

$$\mathbf{X}[k] = \mathbf{W}_{1\to k}\mathbf{X}[0], \tag{3.4}$$

where (3.4) represents the collective dynamics similarly to (3.2) but for the I dimensional case, and $\mathbf{X}[k]$ and $\mathbf{W}_{1\to k}$ are defined as below

$$\mathbf{X}[k] = (\mathbf{x}_1[k] \dots \mathbf{x}_N[k])^T \qquad \mathbf{W}_{1\to k} = \mathbf{W}[k]\mathbf{W}[k-1]\dots\mathbf{W}[1].$$

The weight matrix $\mathbf{W}$ is chosen so that for any initial position matrix $\mathbf{X}[0]$, the state matrix $\mathbf{X}[k]$ converge for $k \to \infty$ and not necessarily to the average vector. This is so because we only seek consensus and among the weight matrix that we will use, only some will lead to a final averaged consensus. Since we will not focus on average consensus the interested reader can find more results and information in [16–19].

Eventually, the conditions for ensuring consensus are:

$$\mathbf{W}\mathbf{1} = \mathbf{1} \tag{3.5}$$

$$\left\|\mathbf{W}[k] - \frac{1}{I}\mathbf{1}\mathbf{1}^T\right\|_2 < 1 \qquad \text{for all} \quad k \tag{3.6}$$

where condition (3.5) has $\mathbf{1}$ as a right eigenvector of $\mathbf{W}$ with associated eigenvalue 1. This condition implies that if consensus is reached the states of the nodes will not change. And condition (3.6) ensures that the algorithm will be stable if the two norm of the equation stays below 1 for all the iterations.

For a general case, where we apply the consensus algorithm (3.1) to a connected graph with I agents and $0 < \alpha < \frac{1}{\Delta}$ with Δ the maximum degree of the graph, if $\mathbf{W}$ fulfills condition (3.5) and condition (3.6), then the following properties hold:

- Consensus is reached for all initial states.
- The consensus point is $\tilde{\mathbf{x}} = \sum_i w_i \mathbf{x_i}(0)$ where the weights w_i are positive and $\sum_i w_i = 1$.
- If $\mathbf{W}$ is double stochastic an averaged consensus is reached and $\tilde{\mathbf{x}} = (\sum_i \mathbf{x_i}(0))/I$.

3.5 Weight Design

There are different schemes feasible to be applied to the construction of the consensus algorithms weight matrix. They have different properties and performance, being specially important that not all of them can reach averaged consensus. Regarding to the performance, the speed of convergence and the ability of staying connected are the most important and will be shown in Section 5. Furthermore, the weight matrix can be symmetric or non-symmetric, having this an impact on the way consensus is achieved. We show here some of the weight matrix designs we have work with.

3.5.1 Structure and Meaning

A weight matrix with size $I \times I$ represents the dynamics of the nodes and determine the direction of movement in every step of the consensus algorithm. Since the position and the degree of the nodes will change over the iterations, the weight matrix may also have different values in every iteration.

Now let us assume that, as explained in Section 3.2 and in Section 3.4, the weight matrix $\mathbf{W}$ is stochastic, at least the sum of the rows is one. This property will be necessary for the rest of the explanation.

Let us focus now in the node 1 and its corresponding row 1. If w_{11} is 1, that would mean that the node will remain in the same position during at least one iteration, that is because the rest of the w_{1j} would be 0. Consequently, it is easy to notice that the entries of the weight matrix are $w_{ij} \in [0,1]$ where $w_{ij} = 0$ implies that the corresponding node j has no effect over i and $w_{ij} = 1$ implies that node j is the only node affecting i.

Therefore a weight matrix with diagonal entries close to 1 would have a really slow convergence while the opposite would occur if these entries were close to 0.

In the special case of the weight matrix for control formation, some entries will result to be negative. It can occur that a node is closer than the agreed distance d with which the nodes have to be separated, then it is necessary a repulsive force. This is the effect of the negative entries.

3.5.2 Symmetric and Non-Symmetric Weight Matrices

Getting consensus means that the weight matrix $\mathbf{W}$ must be at least row stochastic, as explained in Section 3.4. Therefore the matrix $\mathbf{W}$ must be:

- If it is non symmetric then it is only row stochastic, all its rows sum up to one like in condition (3.5) and the system can reach consensus.
- If it is symmetric then all its columns and rows sum up to one. To be column stochastic implies that the average of the states of the nodes remain unchanged,

  $$\mathbf{1}^T\mathbf{W} = \mathbf{1}^T,$$

  then combining both conditions lets averaged consensus to be achieved.

Attending to these properties, having symmetric matrices is interesting but not necessary if reaching an averaged consensus is not required.

3.5.3 Traditional Designs

- Constant weights

  Here we present the first weight matrix approach, the constant weights design (CW). We construct this matrix as in [3]. It is the simplest design and in it all edge weights

  $$w_{ij} \quad \forall \quad j \neq i \qquad \text{and} \qquad (i,j) \quad \in \quad \mathcal{E}$$

  are equal to α. The value of $\alpha \in (0, \frac{1}{\Delta})$ affects the speed of convergence and high values would increase it, although there is a trade-off with the network connectivity. Then the entries of $\mathbf{W}_{\text{CW}}$ are represented as here,

  $$[\mathbf{W}_{\text{CW}}]_{ij} = \begin{cases} \alpha & \text{if } i \in \mathcal{N}_j \\ 1 - d(i)\alpha & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

  where $d(i)$ is the degree of the node i. The construction of the weight matrix is alternatively as below,

  $$\mathbf{W}_{\text{CW}} = \mathbf{I} - \alpha\mathbf{L},$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{L}$ is the Laplacian matrix of the graph. We see from the construction of the weight matrix how the entries of the row are $d(i)+1-d(i)$ so the sum is always 1. Then the $\mathbf{W}_{\text{CW}}$ matrix is always row stochastic, and since the Laplacian of an undirected graph has a symmetric structure, $\mathbf{W}_{\text{CW}}$ will be double stochastic. Consequently, this method achieves averaged consensus.

- Metropolis-Hasting weights

  The second method presented is the Metropolis-Hastings (MH). We followed the design presented in the works [5, 20], where is also possible to find more relevant information about this weight matrix. The most remarkable property is that is the weight matrix in this thesis with the faster convergence. Analyzing the following matrix construction,

$$[\mathbf{W}_{\text{MH}}]_{ij} = \begin{cases} \frac{1}{\max\{d_i, d_j\}} & \text{if } i \in \mathcal{N}_j \\ 1 - \sum_{i \neq j} [\mathbf{W}_{\text{MH}}]_{ij} & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

is possible to appreciate how here the distribution of the nodes in the network is taken in consideration. As said, for high values of the entries of the weight matrix the nodes move faster, but on the other hand this values cannot be over $\frac{1}{\Delta}$. This problem is solved in the constant weight matrix by using always the maximum degree of the whole network, however in the MH case it is used the degree of each node. This leads to have faster convergence because nodes with lower number of neighbors will not be penalized in their speed of movement by the node with the higher degree of the net.

This weight matrix is also symmetric thanks to the method of construction and since the graph is undirected the neighboring nodes have the same degree no matter from which side of the edge we would approach. Recall that the self-weights are chosen so the sum of the weights at each node is 1. Hence averaged consensus can be achieved.

- Rendezvous

  This weight matrix is constructed following the study made by Carlos H. Caicedo-Núñez and Miloš Žefran [4]. As it possible to see in (3.7), the scheme is similar to the constant weight matrix,

$$\mathbf{W}_{\text{RV}} = \mathbf{I} - \alpha(\mathbf{I} - \tilde{\mathbf{A}}), \tag{3.7}$$

  where the constant α here also affects the convergence speed. For RV α is defined with the following boundaries:

$$\alpha \in (0, 1)$$

  The design of the adjacency matrix $[\tilde{\mathbf{A}}]$ is as follows,

$$[\tilde{\mathbf{A}}]_{ij} > 0 \quad \text{if} \quad [\mathbf{A}]_{ij} = 1 \tag{3.8}$$

  such that

$$\sum_i [\tilde{\mathbf{A}}]_{ij} = 1 \tag{3.9}$$

  where it is not necessary to have a symmetric structure. Due to (3.8) the $\mathbf{W}_{\text{RV}}$ matrix will not necessarily be double stochastic. However the stochasticity is ensured with (3.9), as a result with $\mathbf{W}_{\text{RV}}$ is possible to reach consensus. The direct consequence of this new conditions is that the consensus is not agreed in the average point.

3.5.4 Novel Design

- Angular weights

  The problem of the connectivity encouraged the decision of creating a design which would favour the weights w_{ij} of the nodes angularly more isolated. For this new approach the weights are built as the sum of the two smallest angles formed by the vector $i \to j$ for the case of the entry w_{ij}. In example (3.10) is easy to appreciate how the weights $[w]_{ij}$ are computed for the graph in Fig. 3.1.

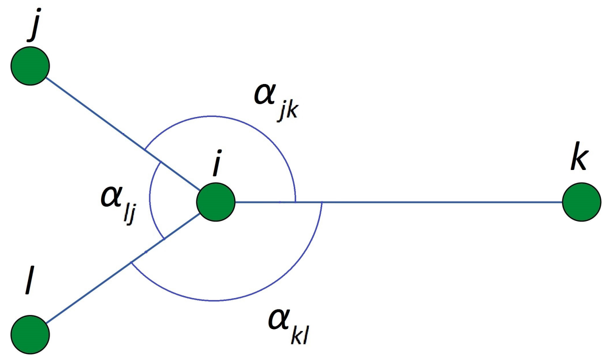


Figure 3.1: *Angular weights*

$$w_{ij} \sim \frac{\alpha_{jk} + \alpha_{lj}}{2} \tag{3.10}$$

The process followed for calculating the weights is the next:

(i) Creating a matrix with the vectors between all the connected nodes.

(ii) Using the matrix of the step 1 was possible to create a new matrix for storing this time the angles between connected nodes.

(iii) Sort the angles and take only the two smallest for each case.

(iv) Eventually, is obtained a matrix $I \times I$ with the coefficients for each edge. This matrix will be called coefficient matrix $\mathbf{C}$. These coefficients sum to 2π in each row, but are normalized to obtain a row stochastic matrix $\mathbf{W}$.

In Fig. 3.2 and Fig. 3.3 the effect of the new weight design can be appreciated, it is seen that the trajectory of the nodes is quite different. In the case of the traditional methods the node i in the middle would move towards the position

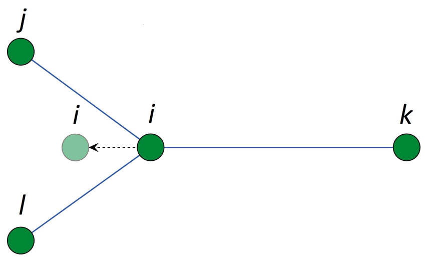


Figure 3.2: *Traditional trajectory*

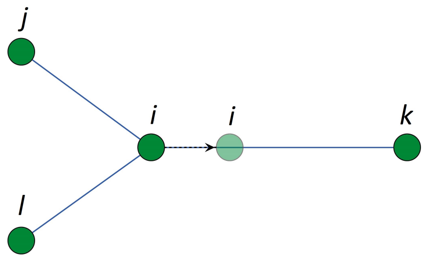


Figure 3.3: *Angles trajectory*

where there is a higher concentration of neighbors, while for the angular design the new position of the node i moves to the more angular isolated neighbor. This effect leads to avoid the network disconnection.

Finally, some other important features are that this design is non symmetric, thus only row stochastic, and what is more, the transmission of state information is the same as with the other weight designs since only the node position is required. The main drawback of the angular weights is the computational cost.

- Modifications of angular weights
  - Symmetric angular weights

    It is possible to symmetrice the angular weight design by means of the coefficient matrix $\mathbf{C}$. The process is as in (3.11).

$$\mathbf{C}_{sym} = \frac{\mathbf{C} + \mathbf{C}^T}{2} \tag{3.11}$$

    When the weight matrix is created from these new symmetric coefficients the result is a symmetric matrix and therefore double stochastic.

  - Distance-Angular weights

    In this case two different coefficient matrices are used in order to include the effect of the distance. Let us $\mathbf{C}_{ang}$ be the angular coefficients and $\mathbf{C}_{dist}$ be the distance coefficients. Then $\mathbf{C}_{dist}$ is created dividing the matrix $\mathbf{D}$ containing the distance between every pair of nodes by the maximum distance. Finally $\mathbf{C}$ is the multiplication element by element of $\mathbf{C}_{ang}$ and $\mathbf{C}_{dist}$.

- Line weights

  This is a special case of matrix which has been developed for one dimensional scenarios. As said, we will mainly focus on two dimension random scenarios, nevertheless we will also study some special distributions such as regular grids and both regular and random one dimensional scenarios.

  For the design of this new weight matrix we have exploited a characteristic which only one dimensional scenarios have, all the agents will be contained in a line and there will always be two nodes, one at each side, which will define the limits of the node distribution. Therefore the idea is shrink the set of nodes starting from the boundaries and keeping the nodes in the middle fixed. To be more precise, the value of the weights will be as below,

$$[\mathbf{W}_{\mathrm{L}}]_{ij} = \begin{cases} \frac{1}{\max\{d_i, d_j\}} & \text{if } i \in \mathcal{N}_j \quad \text{and} \quad i \in \mathcal{B}(\mathcal{G}) \\ 1 - \sum_{i \neq j} [\mathbf{W}_{\mathrm{L}}]_{ij} & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

  where $\mathcal{B}(\mathcal{G})$ represents the nodes on the boundary of $\mathcal{G}$, or in other words, the nodes which only have neighbors on one side.

  Analyzing the structure of the weight matrix, we realize that 3.5 is fulfilled and so will be stochastic. Condition (3.6) is also accomplished, thus the weight matrix will be stable.

We expect that this procedure will let us avoid the disconnection of the network and achieve consensus in a point near but not by its own nature the average.

4 Modifications of Rendezvous

In this chapter, we present some modifications for the RV consensus method. The objective of these changes is to improve the rate of success reaching rendezvous and the convergence properties, or in other words, make the nodes meet faster and ensure that all nodes meet at one point. To achieve these goals, several extensions have been developed. For instance controlling the dynamics, adding delays or reorganizing the network before starting the consensus algorithms.

4.1 Relaying

The first modification is based on relaying information. The idea came in response of the need of obtaining a larger connectivity in order to increase the probability of achieving rendezvous. Relaying creates virtual edges between so far not connected nodes and therefore increases the connectivity. Hence, each node gains more knowledge of the environment and this prevents the agents to move to positions that cause the graph to get disconnected. Another option is to introduce a delay before moving to gather more information, which is discussed in the next section.

Because of relaying the adjacency matrix will change, but strictly speaking this new adjacency matrix will not model the real graph because some edges will appear as a consequence of the relaying. Therefore, the adjacency matrix is virtual and is constructed as,

$$\mathbf{A}_v = \left\{ \left(\sum_{k=1}^{N} \mathbf{A}^k \right) > 0 \right\} - \mathbf{I} \tag{4.1}$$

where $\mathbf{A}_v$ is the virtual adjacency matrix and N is the maximum delay. The operator used in (4.1) is defined as

$$\{\mathbf{M} > 0\}_{ij} = \begin{cases} 1 & \text{if} \quad [\mathbf{M}]_{ij} > 0 \\ 0 & \text{else.} \end{cases}$$

The meaning of ijth entry of

$$\sum_{k=1}^{N} \mathbf{A}^k$$

is the number of different paths from i to j when we can hop at most k times. For understanding why we need the sum in (4.1), we illustrate with a simple example in Fig. 4.1. We see all the possible paths for $N = 2$. If we do not use the sum, node 1 cannot reach 2 and therefore there is no path between them. This is because $\mathbf{A}^k$ represents the number of possible ways to reach a node from any other when we hop strictly k times. In our example it is obvious that, for $k = 2$ it is impossible to go from the red node to the blue and stay, it can only reach its initial position and the green node. So the solution is to sum up as in (4.1) and to make it more clear the structure of $\mathbf{A}$ and $\mathbf{A}^2$ for the example in Fig. 4.1 is the next.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad \mathbf{A}^2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

It is evident that for the graph in Fig. 4.1 the equation (4.1) produces a fully connected virtual adjacency matrix when $k \geq 2$.

We have seen that the zero pattern of $\mathbf{A}_v$ will be different from $\mathbf{A}$ for $k > 1$, and since there will be more edges, which is equal to less zeros in $\mathbf{A}_v$ than in $\mathbf{A}$, we obtain a higher connectivity.

In order to apply relaying it is necessary to move only every certain number of iterations. As a result the agents can exchange state information with second or higher order neighbors and decide a more precise movement because they have more information. As a consequence, all neighbors have to wait for the message of node i before they can move, and this is time consuming. Moreover, in each iteration all $|\mathcal{N}(i)|$ neighbors have to additionally spread the position of node i. Therefore the communication load increases significantly

$$\left(\frac{1 + |\mathcal{N}(i)|}{1}\right).$$

Despite the drawbacks, it is important to notice that every k first iteration, when the nodes do not move, are necessary for computing (4.1). The reason why we say that the movement will be more accurate is because the RV weight matrix will be created using the zero pattern of $\mathbf{A}_v$, which we recall that contains more non-zero entries. Eventually

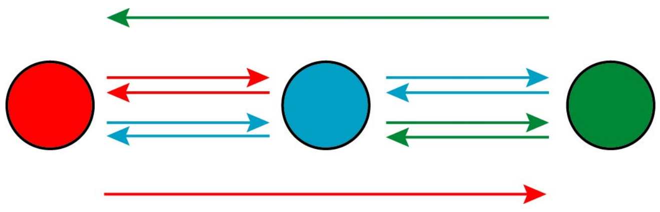
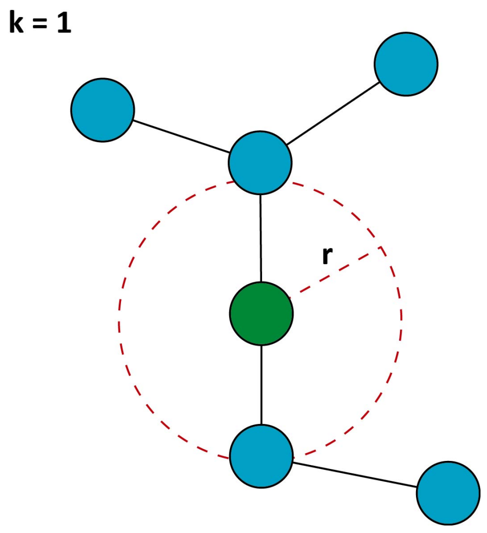


Figure 4.1: *Feasible paths*

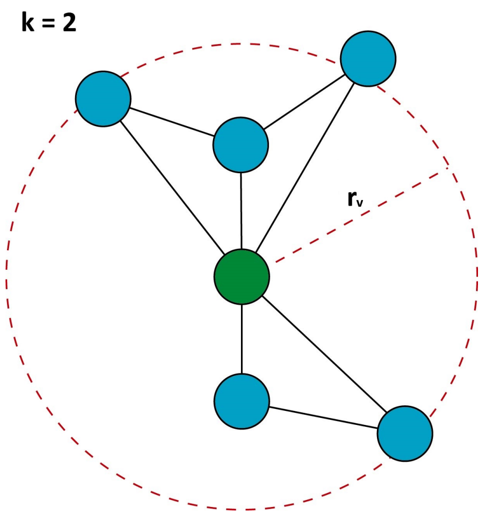
everything reduces to the idea of having more information for taking better decisions.

The relaying effect is seen in Fig. 4.2. It is seen that the final consequence of relaying is the appearance of a virtual connectivity range r_v which is $r_v \leq 2r$, where r is the connectivity range. We see that r_v cannot be $2r$ in random geometric topologies because this would need an infinite number of nodes at a distance equal to the connectivity range. However, in the case of some regular network topologies, this condition in fulfilled, therefore $r_v = 2r$. In Fig. 4.3 we see a non-regular and a regular distribution where the connectivity range for each node is $r_i = r$ for all i. We see that node 1 can not reach the node 5 because the neighbor node in its direction is at a distance smaller than the connectivity range in Fig. 4.3(a). Therefore, despite node 5 is at a distance $2r$ from 1 it cannot be reached. In the other example illustrated in 4.3(b) all nodes at a distance $2r$ from 1 can be reached, since all the neighbor nodes are distributed at the same distance r from 1.

The performance of the relaying is considered in Section 5. We expect to keep the networks connected and achieve consensus in one point more often than with the raw RV algorithm.

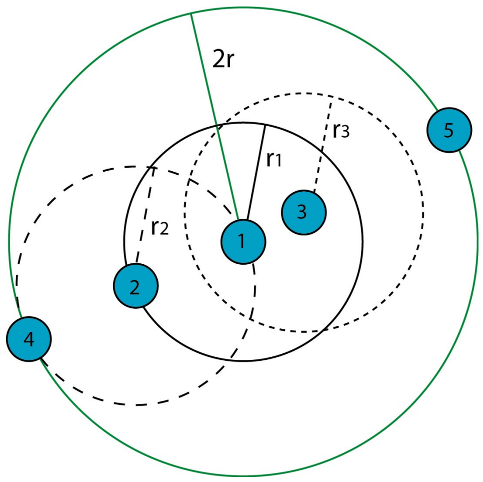


(a) *Before relaying*

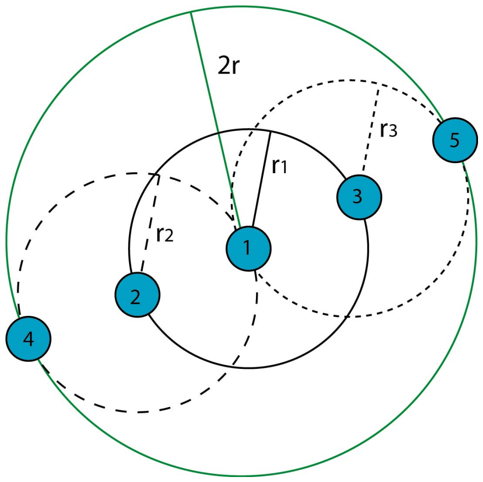


(b) *After relaying*

Figure 4.2: *Relaying*



(a) *Non-regular distribution*



(b) *Regular distribution*

Figure 4.3: *Relaying*

4.2 Delay

The second modification introduces a delay in the movement in order to virtually increase the connectivity before making a movement. This approach is slightly different from our relaying method. The delay provides us with extra time that can be used for exchanging more information like in the relaying case, but this time we simply let consensus algorithm run more iterations without letting the nodes move. Since the graph remains constant during the delay, running consensus algorithm d times within the delay, is equivalent to compute the weight matrix to the power of d and apply it once. The update equation for our delaying method, with k as iteration from which we start to apply delay, yields

$$\mathbf{X}[k] = \begin{cases} \mathbf{W}^d\mathbf{X}[k-1] & \text{if} \quad k - d\lfloor \frac{k}{d} \rfloor = 0 \\ \mathbf{X}[k-1] & \text{else,} \end{cases} \tag{4.2}$$

where

$$\mathbf{W}^d = \mathbf{W}_{k-d\to k} = \mathbf{W}[k-d]\ldots\mathbf{W}[k-1]\mathbf{W}[k],$$

and also,

$$\mathbf{W}^d = \mathbf{W}[k-d]^d \qquad \text{or} \qquad \mathbf{W}^d = \mathbf{W}[k]^d,$$

because the weight matrix does not change while we apply the delay. For instance, if we start from $k = 1$ and $d = 3$, we will not update the position until the iteration number 4 being (4.2) as follows

$$\mathbf{X}[k+d] = \mathbf{W}^d\mathbf{X}[k],$$

$$\mathbf{X}[4] = \mathbf{W}^3\mathbf{X}[1],$$

and therefore the nodes will not move before that, and consequently during these iterations no component can get disconnected if at $k = 1$ was not. Note that (4.2) can be written also as follows

$$\mathbf{X}[4] = \mathbf{W}^3\mathbf{X}[2],$$

because $\mathbf{X}[1] = \mathbf{X}[2]$, since the nodes do not move during the two first iterations.

Ideally, if the we have a connected graph $\mathcal{G}$ and $d \to \infty$ then (4.2) is as below

$$\mathbf{X}_{\text{cons}} = \mathbf{W}^{\infty}\mathbf{X},$$

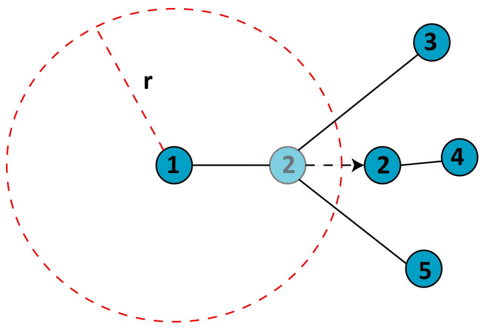
and $\mathbf{X}_{\text{cons}}$ is a point where all the nodes would converge in just one movement if the speed of movement is not constrained. Furthermore, if the weight matrix design is double stochastic,

$$\mathbf{X}_{\text{cons}} = \tilde{\mathbf{X}},$$

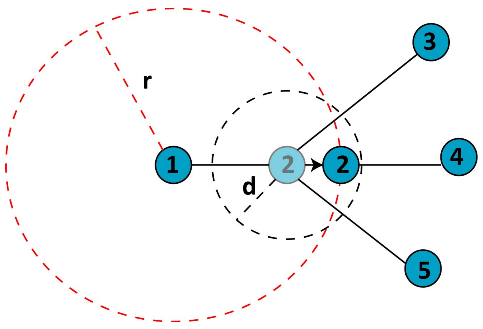
where $\tilde{\mathbf{X}}$ is the averaged sum of the initial positions. The drawback is of course that $d \to \infty$ is practically infeasible. Hence the goal is to find a suitable d for making the network converge and at the same time a small d is preferable because we will be adding a smaller delay.

Computing the consensus matrix many times before moving creates a smoothing effect on the matrix, in other words it reduces the differences between the weights and creates a virtual range augmentation because it changes the zero pattern of the weight matrix by adding non-zero entries in positions which correspond to unconnected nodes. This reduces the chance for the network of being disconnected because the virtual weight matrix is created as if the nodes knew the position of more neighbors.

In the following we briefly compare and analyze the differences between the relaying and delaying approach. The first thing we should recall is that with relaying we do not do any processing while we relay state information, at the same time, with delaying we



(a) *Without thresholding*



(b) *With thresholding*

Figure 4.4: *Thresholding*

continuously process and share data with the neighbors. Hence, the first modification affects the adjacency matrix, whereas the second changes the weight matrix. Nevertheless, both have similar effects, they modify the zero pattern of $\mathbf{A}$ and $\mathbf{W}$, respectively, and in both cases the number of zero entries in the matrix is reduced. For both methods of course the network stays connected if the nodes do not move. The last important consideration is that we expect the delay method to work better, because it not only applies a delay for gathering more information, but also evolves the weight matrix to a more advance state.

4.3 Thresholding

This modification changes the perspective from which the connectivity problem has been addressed before. The motivation behind this method is that slower nodes should decrease the chance of getting disconnected. Hence, we will study the effect of constraining the traveled distance and thus speed. Fig. 4.4 illustrates the expected behavior. It shows how in this case limiting the traveled distance reduces the negative effect which areas with a relatively large number of nodes produces. In this specific scenario the central node 2 moves faster towards the area with more nodes because the overall attraction they cause is bigger than the isolated node 1, whereas node 1 moves slower because only has the node 2 under its scope. We see in Fig. 4.4(b) that node 2 stays within the scope of 1 when thresholding is applied. In Fig. 4.4(a), however, we do not control the traveled distance and the graph gets disconnected.

First of all, we need to assign threshold value, which we denote t. Then we apply the following steps,

- We start conventional consensus algorithm using the RV method. So we obtain $\mathbf{W}_{\mathrm{RV}}$ and compute the new position for every node.
- Next we calculate the Euclidean distance walked by each node. The Euclidean distance for a two dimensional scenario is as in Fig. 4.5, where p_i and q_i are the initial and final positions respectively for node i. The distance node i travels will be equal to

  $$\mathbf{d}_E = ||q_i - p_i||_2,$$

  where both p_i and q_i are defined by their x and y coordinates since we work within a 2D scenario. Therefore d_E is the Euclidean distance covered by node i. More information about the definition used for the Euclidean distance can be found in [21].
- Finally we compare with the threshold t and if

  $$[\mathbf{d}_E]_i > t,$$

  then node i would travel further than the threshold distance t, and is therefore forced to move only distance t at maximum in the same direction.
- Once all the nodes have computed their new positions meeting the criteria, they can move.

It is clear that this modification will have a negative effect on the convergence properties because we are limiting the speed of the nodes. Nevertheless, we expect that in the end

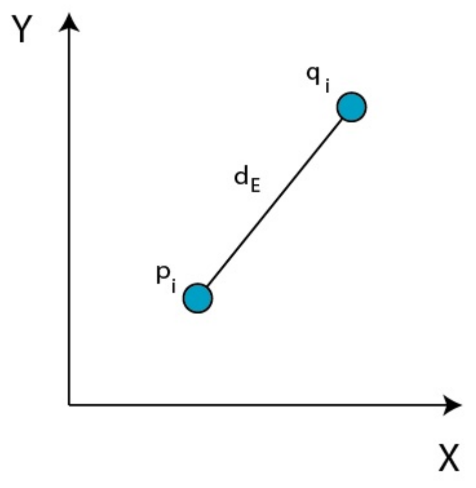


Figure 4.5: *Distance travelled by node i in one iteration*

it will help to keep the network connected.

4.4 Dispersion

The last modification does not apply control or changes of the Rendezvous algorithm like the others, but a new algorithm prior rendezvous is applied to the network in order to get a more even distribution of the nodes. This formation control algorithm is based on ideas presented in [1, 22–26]. In [27] information about flocking, another interesting application, which make a cluster of nodes move in the same direction with the same speed, can be found. We will not develop any algorithm for flocking in this work, however it could be an interesting topic for a further extension.

For the construction of the dispersion algorithm we use a symmetric matrix containing the Euclidean distance between the nodes. Obviously, the diagonal elements are zero since it bears the distance between a node and itself. Hereafter we call this matrix $\mathbf{D_F}$ and impose the zero pattern of $\mathbf{A}$, i.e., we set the entries of $\mathbf{D_F}$ corresponding to nodes that are not connected to zero. Then we choose a distance d as the aim of the separation between neighboring nodes. The consensus weight matrix for dispersion is

$$\mathbf{W}' = \mathbf{D_F} - d\mathbf{1}\mathbf{1}^T + d\mathbf{I},$$

where the diagonal contains zeros. To guarantee the row stochastic property, we add a diagonal matrix which makes the sum of the row 1,

$$\mathbf{W} = \mathbf{W}' + (\mathbf{I} - \mathrm{diag}(\mathbf{W}'\mathbf{1})).$$

If we analyze the structure of the weight matrix we notice that it contains negative entries. These entries correspond to the nodes which are closer than d and therefore need a repulsive effect in order to reach the objective distance between them. This is not enough to avoid clustering as we will explain in more detail later in this section.

Fig. 4.6 shows how a regularly distributed network may look. We can see how the nodes keep a distance which is or tends to d. Ideally all of the nodes are equally spaced because the goal is a regular network distribution.

The dispersion method cope with some issues which make the system unstable or lead to unsatisfactory results. We describe the solutions we have implemented here:

- Oscillations: Having an inappropriate or too large speed of convergence provokes undesired oscillations which makes consensus unreachable. As explained in Section 3.5.1, large diagonal entries cause a slow movement of the nodes. Therefore, before adding the values to the diagonal for making the $\mathbf{W}$ matrix stochastic, we divide the $\mathbf{W}'$ matrix by a constant $c > 1$.

$$\mathbf{W} = \frac{1}{c}\mathbf{W}' + (\mathbf{I} - \frac{1}{c}\mathrm{diag}(\mathbf{W}'\mathbf{1}))$$

- Range: The connectivity range affects the formation of the network. Having a large connectivity range provokes a grouping of nodes in clusters of approximate radius d. The reason why this occurs with high values of connectivity range is because the nodes in the middle of the network pull nodes on the boundary towards them. These nodes on the boundary try to be at a distance d to the nodes in the middle but on the way they drag other nodes which were in the middle. The result are the above mention clusters of nodes. This effect can be counteracted by setting a connectivity range similar to the separation objective d.

- Stucked nodes: Sometimes nodes which are too close and surrounded by many others have difficulties for reaching the separation d. This problem has been addressed by boosting the negative entries of the weight matrix. This means that the nodes are forced to move further and we expect that this will also help to separate the clusters of nodes which may appear in the different scenarios due to the random nature of the initial graph topology.

- Exceeding the limits: Finally we consider the case when nodes try to cross the boundary of the prescribed area of the network. This is very likely for dispersion since it has an spreading effect. Two approaches how to handle this phenomenon are described in the following.
  - The first approach is making the coordinate which steps out of the frame to change its value to the maximum possible. The effect can be seen in Fig. 4.7, where the coordinate x is going out of bounds, but instead we keep the coordinate y and fix x to be the maximum possible value.
  - The second approach is assuming that we have a solid surface, so the node will bounce. This means that the node will continue its way by only changing the sign of direction in the axis which is reaching the limit, as can be seen

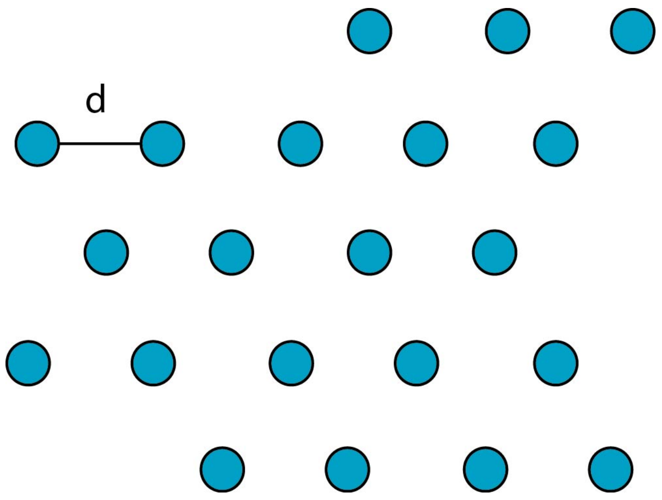


Figure 4.6: *Network after dispersion*

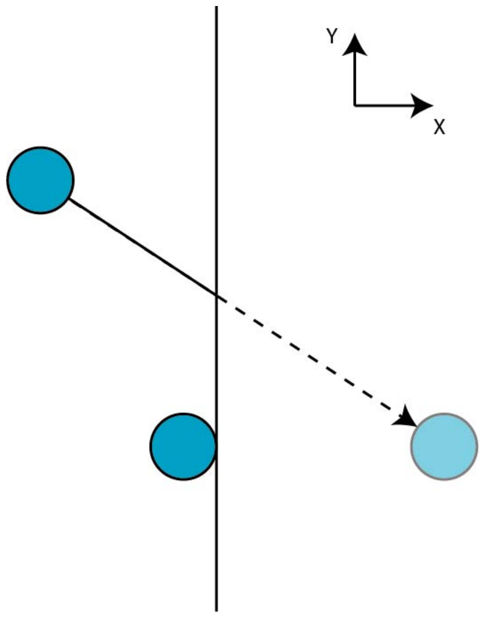


Figure 4.7: *Limiting*

in Fig. 4.8.

In Section 5 we will see the performance of RV and AW in networks where we applied formation control algorithm. RV will present poor results whereas AW achieves an outstanding successful rate.

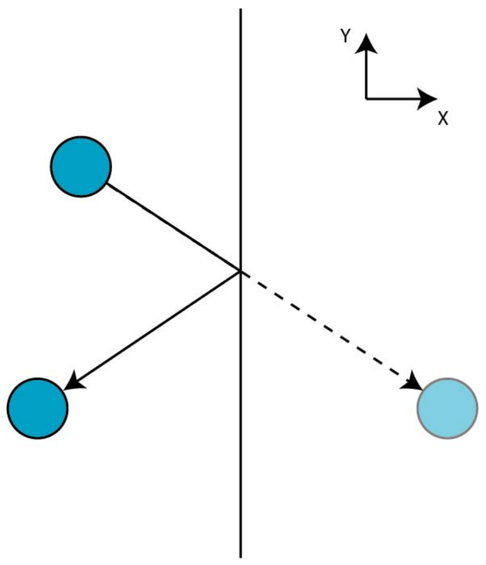


Figure 4.8: *Bounce*

5 Simulation Results

In this chapter we show the results of the simulations obtained. The objective is first to compare the different weight matrices, and then to check how the proposed modifications affect the performance. Then we will test the behavior of the consensus algorithms under special circumstances like regular networks or one dimensional scenarios.

5.1 Simulation Setup

Before showing the results we explain the settings used. Some of the parameters change along the different simulations because they depend on the needs or they are used for checking how their variation affect the performance. These are for instance the step-size α, the number of agents I, the number of iterations k or the communication range r. For us α is the step-size. It defines how fast the nodes can move and therefore affects the speed of convergence, α stays always within the limits described in Sections 3.2 and 3.4. The number of agents I, the number of iterations k, with special attention to the maximum number of iterations k_{max}, or the communication range r, are used for tuning the algorithm and testing which are the best settings depending on the objectives. Let us define r as the maximum reachable Euclidean distance, following again the definition in the work [21], this yields that the maximum distance d which can be covered is

$$d = |x^2 + y^2| \leq r,$$

where x and y are the distance covered in each coordinate. The communication range is decided using the equation,

$$r = c\sqrt{\frac{\mathcal{A}}{I-1}},$$

where A is the area of the frame, I is the number of nodes and c is a constant which determines the communication scope of the agents. In most of the simulations the number of scenarios used are 1000, however sometimes will vary. We use a large number of random scenarios for averaging and achieving more realistic results. In some subsections we will provide a table with the parameters used.

Other parameters are fixed, in order to unify all the results as much as possible. The area under study is always $\mathcal{A} = [0, 1] \times [0, 1]$. Furthermore, the initial graph is random geometric and always connected at the beginning but in the cases of control formation and the study of regular matrices. In control formation having initially unconnected graphs helps, indeed, for studying if these graphs can get connected.

The mean-squared-error (MSE) is defined as the mean-squared distance of the agents to the average position. The average position is calculated as the sum of the positions of all the nodes and then dividing by the number of nodes, and repeating for each iteration. The process for calculating the MSE starts with the average squared error (ASE) [14],

$$\mathrm{ASE}[k] = \frac{1}{I}\sum_i |\mathbf{x}_i[k] - \bar{\mathbf{s}}[k]|^2, \tag{5.1}$$

where $\bar{\mathbf{s}}[k]$ is the average position at iteration k, and then using the equation (5.1) the MSE is as follows

$$\mathrm{MSE}[k] = \frac{1}{CI}\sum_c\sum_i |\mathbf{x}_{ci}[k] - \bar{\mathbf{s}_c}[k]|^2,$$

where C is the number of scenarios. Therefore the MSE we use is also averaged by the number of scenarios in order to gain accuracy.

5.2 Comparing Weight Matrices

All along this section we are going to explain the results obtained for every different weight matrix design. Furthermore, we will compare the trajectories of the nodes for different weight matrix designs and over the same scenarios, highlighting the differences.

The settings used in this section are shown in Table 5.1.

Parameter	Value
I	100
$k_{\max}$	150
r	0.24
No. of scenarios	1000

Table 5.1: *Parameters Section 5.2*

5.2.1 MSE Performance

In this section, we will use the mean-squared error (MSE) for measuring which weight matrix is better. With the MSE it is possible to check not only if a method converges faster, but also which method is achieving consensus with higher probability because it will result in a lower average MSE. To be more clear, if for the last iteration the normalized number of neighbors is 1 then the network is fully connected and the consensus can be achieved.

In Fig. 5.1 and Fig. 5.2 we see two plots. The first with the evolution of the MSE over the iterations for the four studied weight matrices and the second with the normalized number of neighbors over the iterations.

Now let us focus on Fig. 5.1, where are shown all the scenarios, regardless of whether the graphs are connected. We see how AW outperforms the other methods both in convergence speed and in MSE performance. Among the traditional methods rendezvous is the best because it achieves consensus with a higher probability but MH converges faster. With respect to the number of neighbors, Fig. 5.1 shows a high dependence of the MSE improvement on the number of neighbors.

Fig. 5.2 is similar to Fig. 5.1 but for only connected scenarios, or in other words, scenarios with which in the end we can reach consensus. It is possible to see that only RV and AW can achieve consensus. For the case of RV only a few scenarios converge and we can observe some artifacts in the number of neighbors plot due to this circumstance. Comparing RV and AW we can see that initially RV converges faster but eventually AW achieves better MSE results.

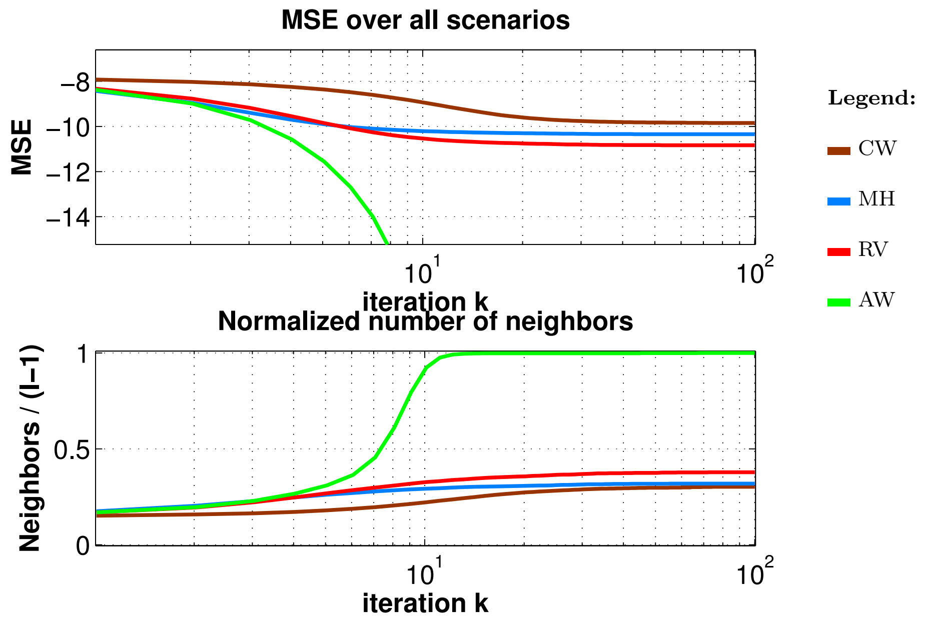


Figure 5.1: *Averaged mean squared error*

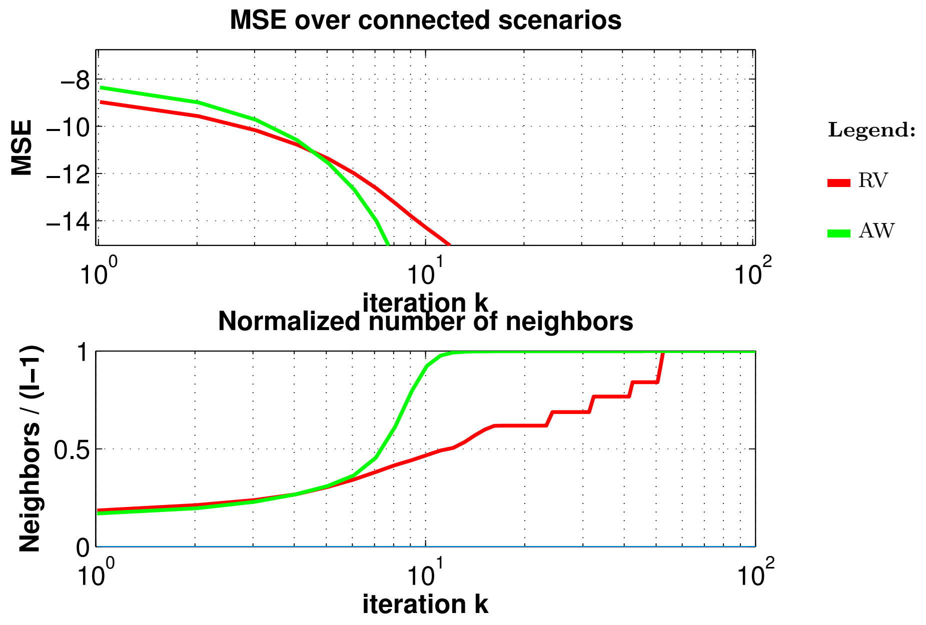


Figure 5.2: *Averaged mean squared error over connected scenarios*

5.2.2 Average Distance Traveled

This section gives some hints about why AW has a better performance. And one possible reason lies in the initial speed of the nodes. First we will study the average traveled distance in general and then moving on to the average distance between nodes. The averaged travelled distance is the sum of distance the nodes move divided by the number of nodes while the average distance between nodes is the sum of the distance between each pair of connected nodes divided by the number of nodes time the number of edges.

Let us start with the average traveled distance, note that we again distinguish all the scenarios and connected scenarios in Fig. 5.3 and Fig. 5.4. In Fig. 5.3 we can realize how AW has a different behavior than the other methods. AW starts with slow movements and increases the speed as long as the connectivity increases. This helps to preserve the structure of the network and avoids early disconnection. On the other hand MH and RV start fast and the isolated nodes get often disconnected. Regarding CW the performance is relatively poor and, despite the slow speed of movement, the network suffers from disconnection. Fig. 5.4 shows again similar results than Fig. 5.3 but for connected scenarios. Again, since only a few scenarios converge for RV, we see some artifacts.

The second approach is the evolution of the distance between nodes. This time only RV and AW are analyzed. In Fig. 5.5 we have considered all the scenarios and also the distance between nodes which may not have and edge between them, while Fig. 5.6 show the distance between nodes which are connected. We can see how the distance between nodes with RV is reduced faster, whereas for AW at the beginning this distance stays almost constant. In Fig. 5.5 there is a saturation effect for RV when measuring the distance between all nodes because there are more scenarios which can not reach consensus. In Fig. 5.6 for the same reason irregular peaks appear for RV between con-

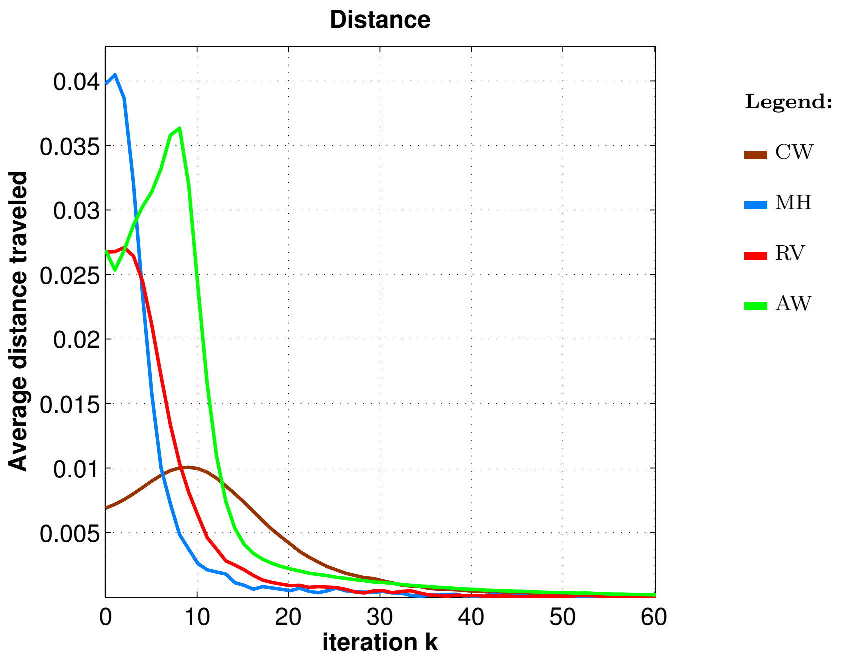


Figure 5.3: *Average traveled distance over connected scenarios*

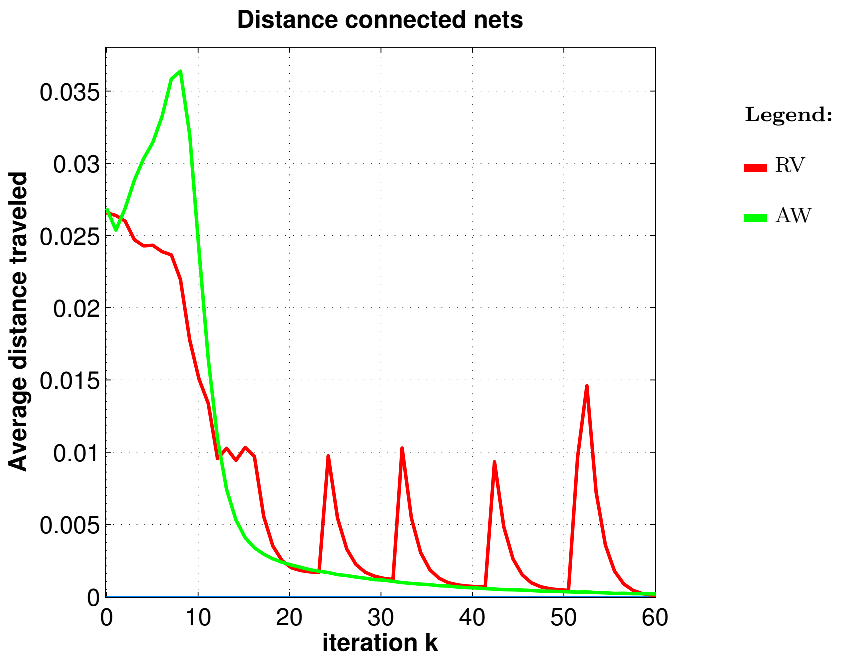


Figure 5.4: *Average traveled distance over all the scenarios*

nected nodes. Fig. 5.5 and Fig. 5.6 support the idea which states that slow initial movement helps to stay connected.

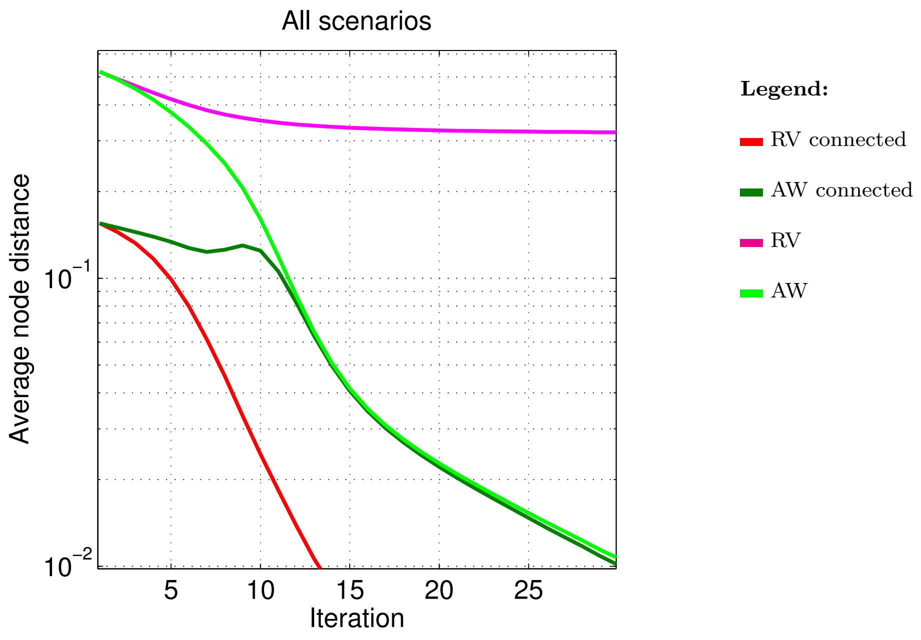


Figure 5.5: *Average node distance over all the scenarios*

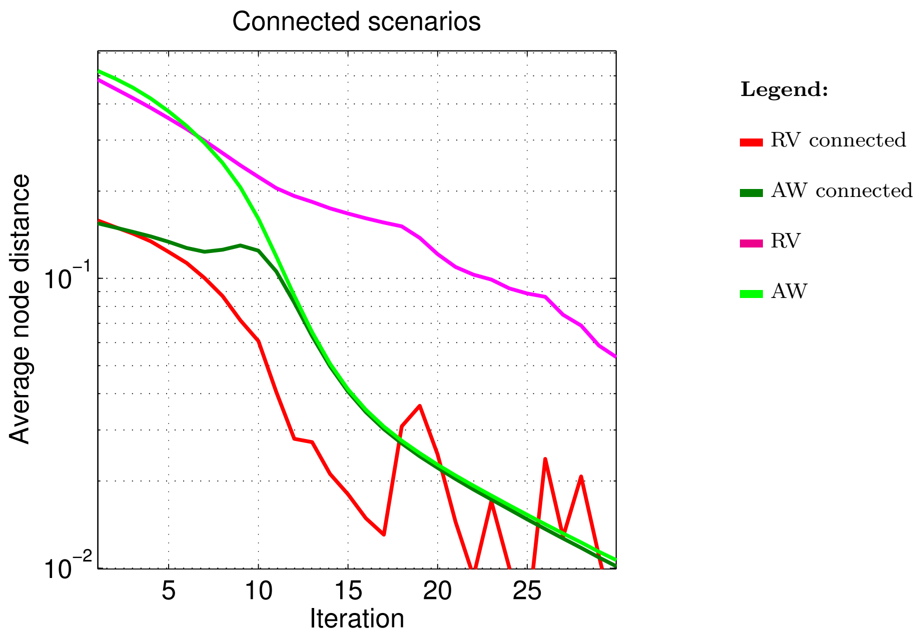


Figure 5.6: *Average node distance over connected scenarios*

5.2.3 Trajectories

Now we are going to analyze how different weight matrices provoke different trajectories of the nodes even when the scenario is the same. We will show first the trajectories for the graph in Fig. 5.7. Fig. 5.8, Fig. 5.9 and Fig. 5.10 show clearly how the traditional

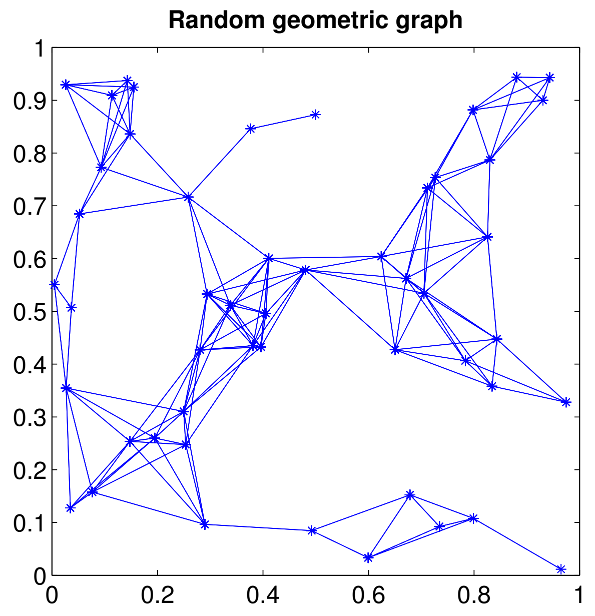


Figure 5.7: *Network under study*

methods tend to converge locally when the connectivity is not high enough. In contrast, in Fig. 5.11 AW keeps the nodes connected and they eventually reach consensus. In Fig. 5.12(a) and Fig. 5.12(b) a comparison between RV (in red) and AW (in green) is depicted. In Fig. 5.12(b) we have made a zoom to see in more detail what occurs in a specific region of the network. For the RV nodes on the upper left corner having more neighbors in their area provoke disconnection with the rest of the network in a few iterations. However, for AW, since the weights are computed with the angles, the structure is preserved and the consensus is achieved. In Fig. 5.13 and Fig. 5.14 are displayed the number of graph elements and its position for the network in Fig. 5.12(a). It is a simple way to check how RV gets 3 local consensus points while AW achieves the objective of general consensus, because the graph elements let us know whether the

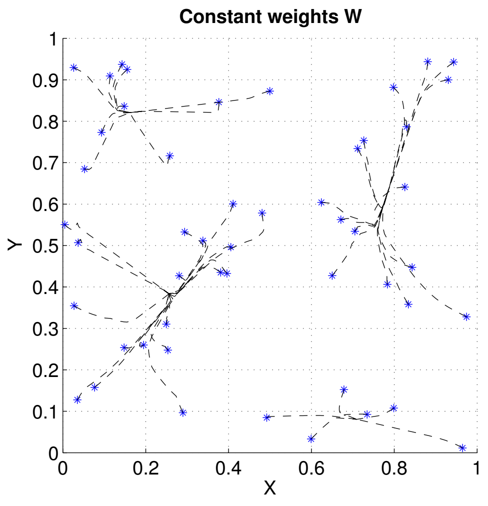


Figure 5.8: *CW trajectory*

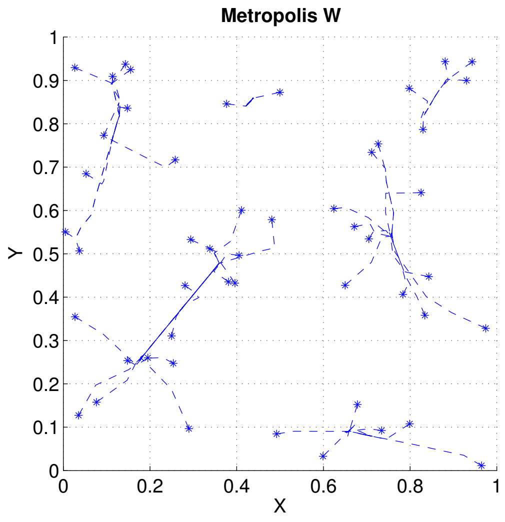


Figure 5.9: *MH trajectory*

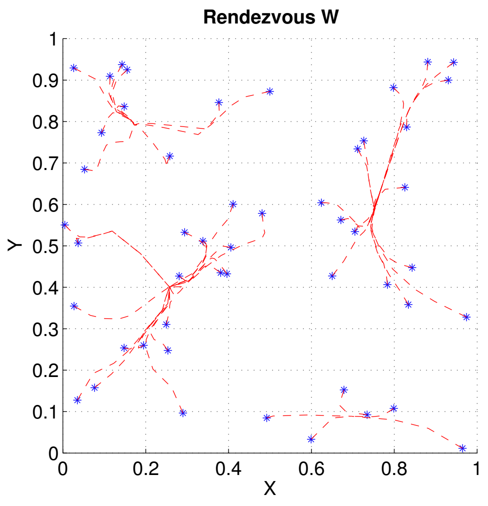


Figure 5.10: *RV trajectory*

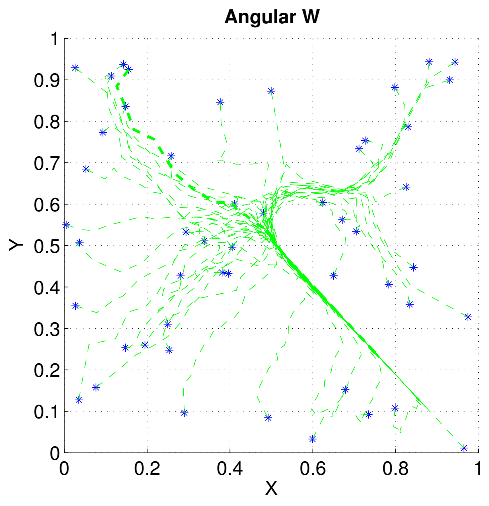
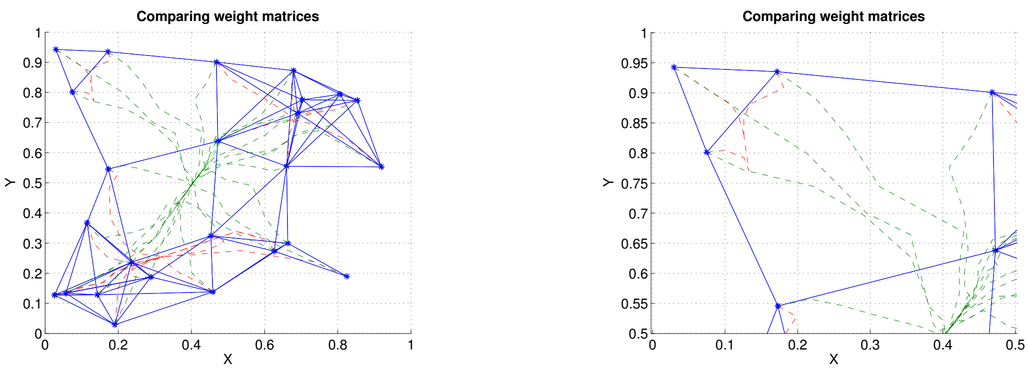


Figure 5.11: *AW trajectory*

network is fully connected or not. The graph elements are the number of independent clusters of nodes that we obtain after running a consensus algorithm over a network for k iterations. Therefore if we only have one graph element the network is fully connected. If we have more than one graph element means that the network is unconnected and the consensus can only be achieved locally. We recall that for this section the networks are initially connected, thus they only have one graph element at the first iteration.



(a) *Whole frame* (b) *Zoomed area*

Figure 5.12: *Comparison between RV and AW*

5.2.4 One Dimensional Scenarios

In this section we will present some results obtained by applying RV and the line weights design over one dimensional scenarios. The creation of a new weight matrix was motivated as an effort for getting better performance than with the classic RV. The setup used can be seen in Table 5.2.

First we compare the number of graph elements in Fig. 5.15 and Fig. 5.16, or in other words if the networks stay connected. In Section 2.1.4 was explained in more detail. Fig. 5.15 shows how RV is affected by the number of neighbors and the range, while in Fig. 5.16 it is possible to see how line weights achieve consensus for all the scenarios under these settings.

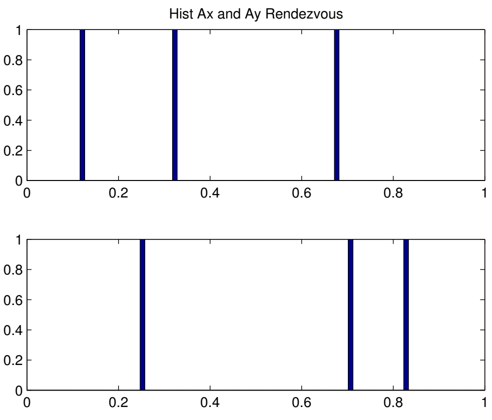


Figure 5.13: *RV histogram*

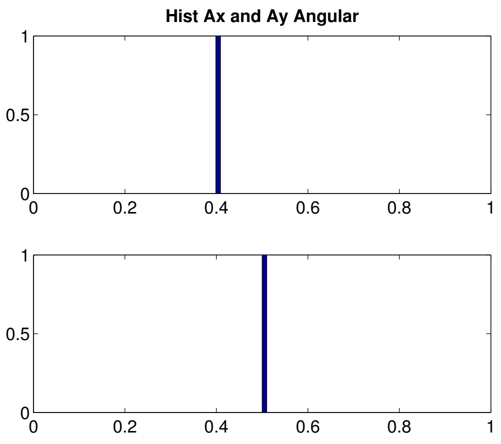


Figure 5.14: *AW histogram*

Parameter	Value
I	50-100
$k_{\max}$	200
r	$0.16 - 0.24$
No. of scenarios	100

Table 5.2: *Parameters Section 5.2.4*

In Fig. 5.17 we compare the speed of convergence of RV and line weights. Whereas line weights is better in terms of consensus rate, we can see how RV has a faster convergence.

5.3 Modifications Of Rendezvous

This section deals with the modifications implemented over the RV algorithm. These modifications are controlling some aspects of the algorithm such as the distance traveled or the frequency of movement. The settings for this section are in Table 5.3. In

Parameter	Value
I	100
$k_{\max}$	150
r	0.24
No. of scenarios	1000

Table 5.3: *Parameters Section 5.3*

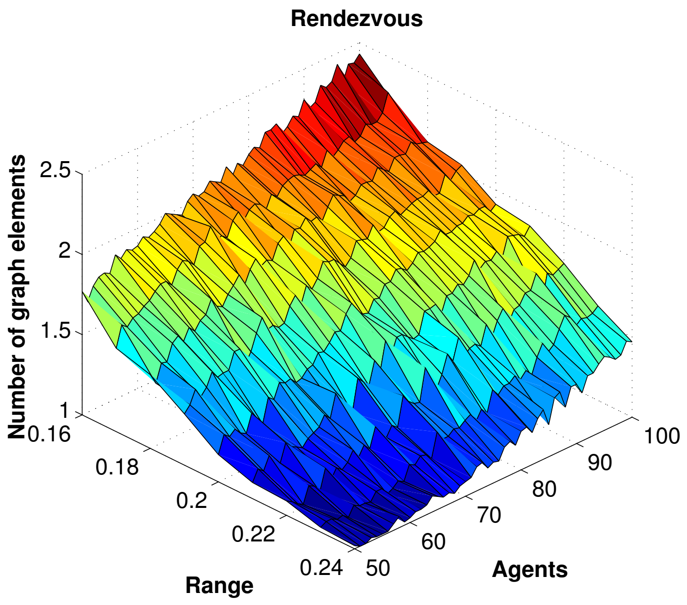


Figure 5.15: *Number of graph elements depending on the connectivity range and the number of nodes*

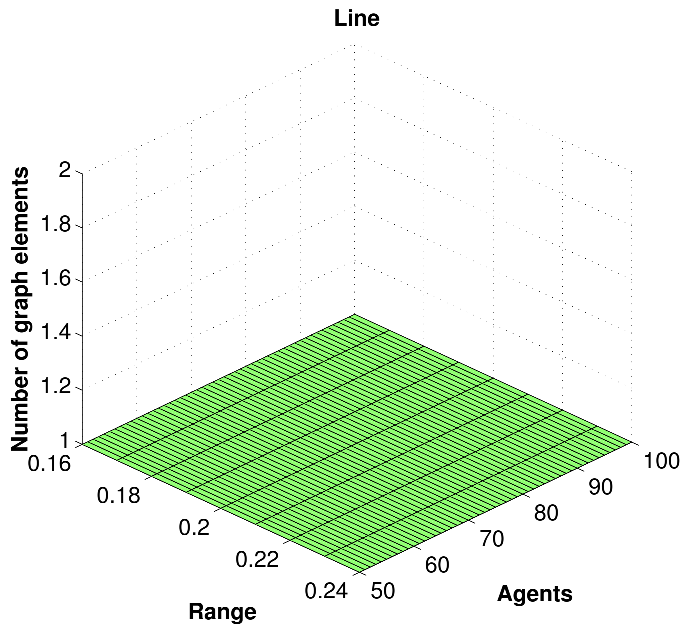


Figure 5.16: *Number of graph elements depending on the connectivity range and the number of nodes*

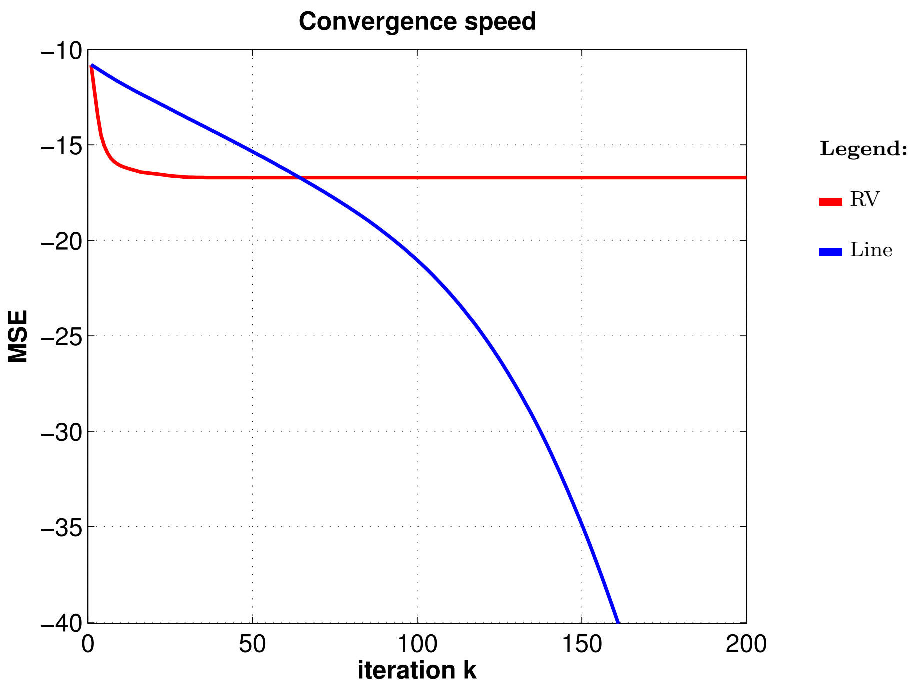


Figure 5.17: *Convergence speed of RV and line weights*

Fig. 5.18 and Fig. 5.19 we compare the evolution of the MSE over 100 iterations of the different modifications. It is clear that for both only connected scenarios and all the scenarios the delayed versions are giving a great performance. On the other side, the methods which add a constraint in the distance traveled are even counter-productive. For the delayed versions not only the MSE is good but also the convergence is faster and also the speed with which the number of neighbors grow.

Regarding the distance traveled, we have decided to show only the threshold method because the delayed versions converge faster and in a reduced number of steps. We see in Fig. 5.20 and Fig. 5.21, that plotting only connected scenarios does not vary the distance traveled but the curve for all the scenarios is smoother because few scenarios converge for the threshold modification.

In Fig. 5.22 appear the percentage of networks achieving rendezvous, as expected the AW and the delayed versions of RV are getting the best results.

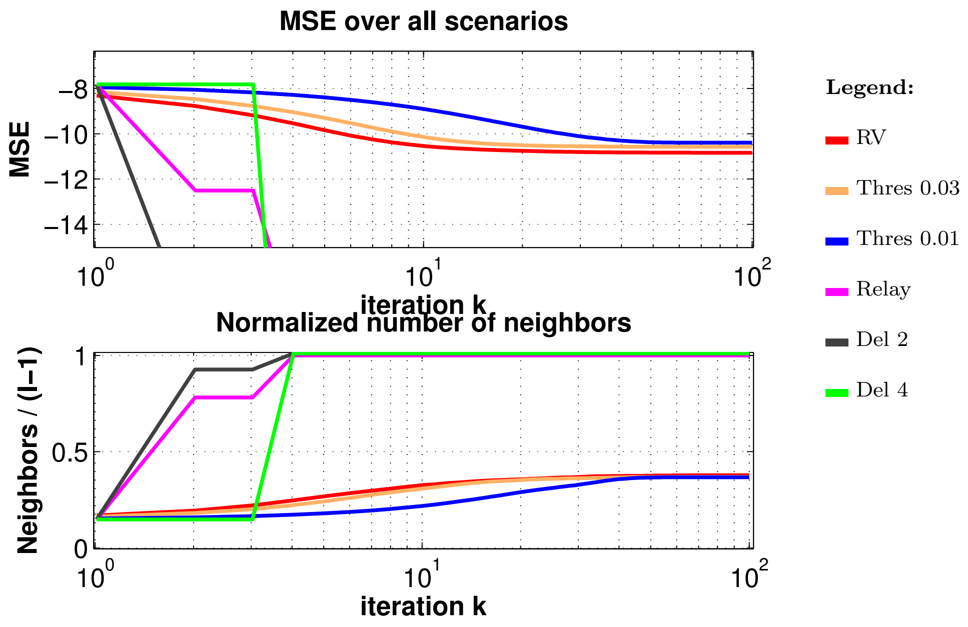


Figure 5.18: *Mean-squared error over all the scenarios*

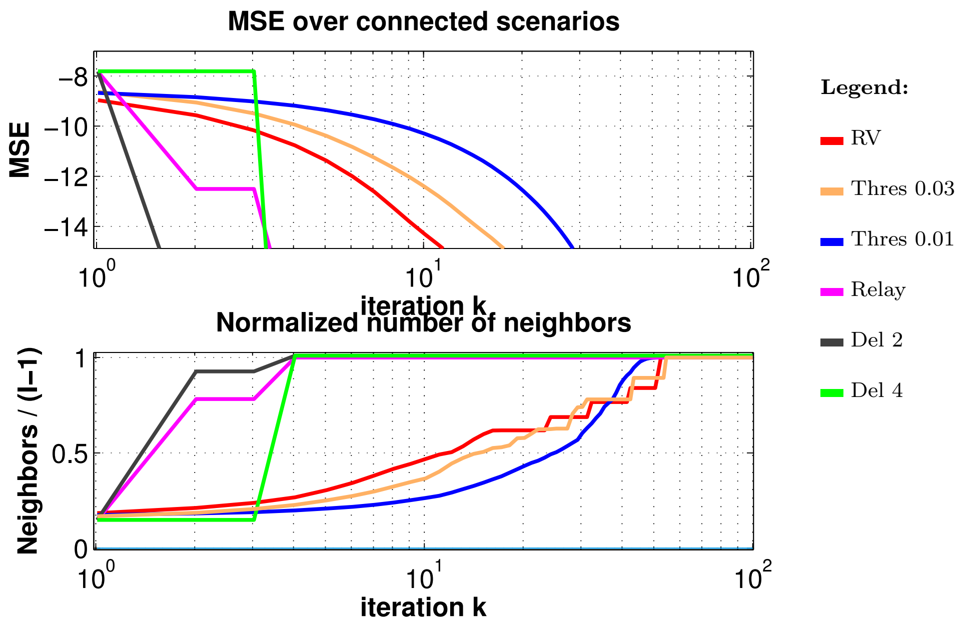


Figure 5.19: *Mean-squared error over connected scenarios*

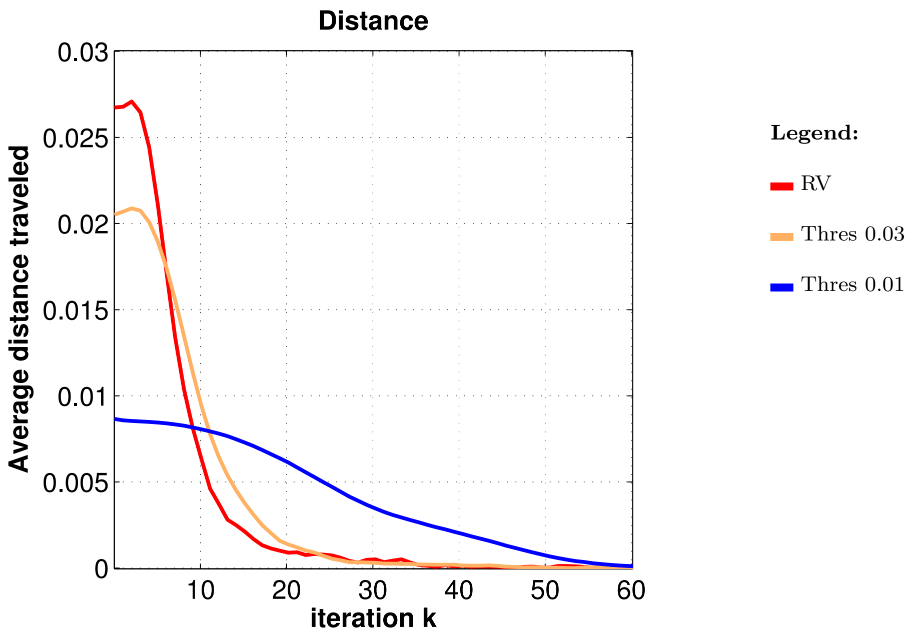


Figure 5.20: *Average traveled distance over all the scenarios*

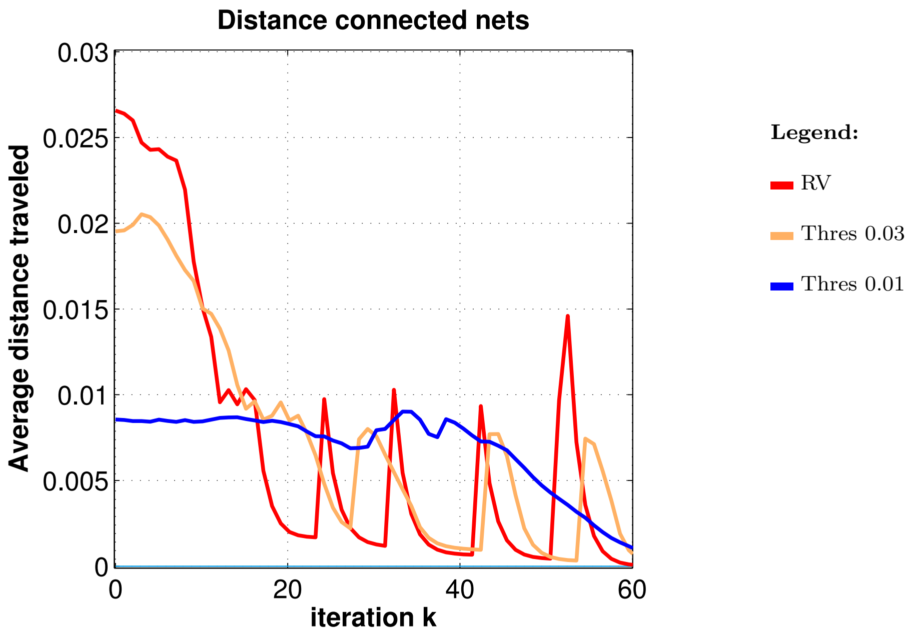


Figure 5.21: *Average traveled distance over connected scenarios*

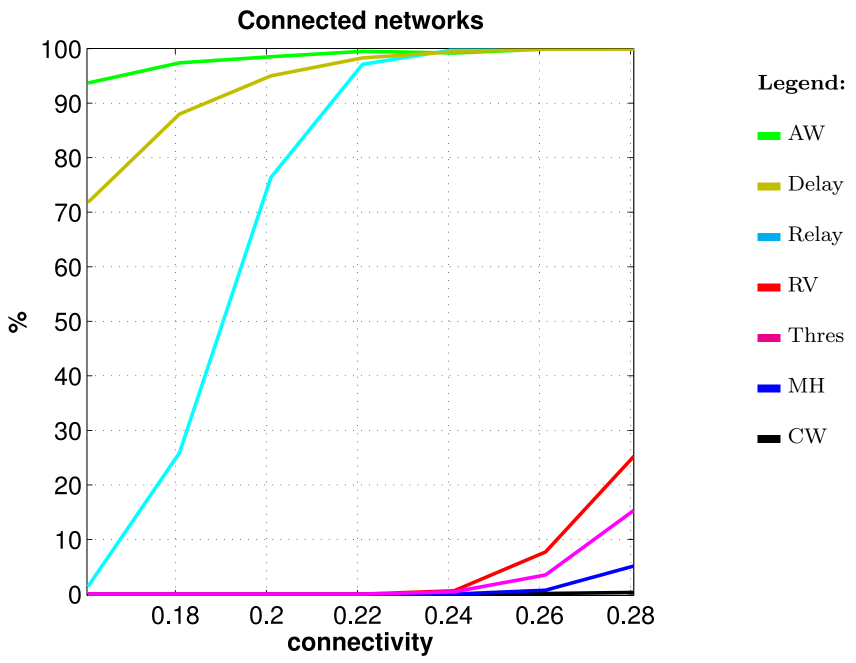


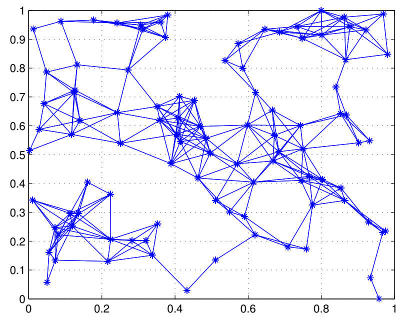
Figure 5.22: *Succesful rendezvous*

5.3.1 Dispersion

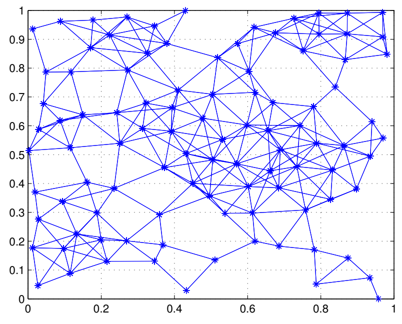
In this section we will discuss the advantages of using dispersion before running the rendezvous algorithm. Despite it is one of the suggested modifications, a different section is used due to its particularities. In first place we need to prove that dispersion is worthy and then in which range because the network we obtain after the dispersion can have different distance d between the nodes. Hence, in this section we analyze which values of d are useful for our consensus purpose.

The settings used are the same that for the rest of the section, with the exception of the connectivity range, which is used as a variable with the values $r = \{0.16, 0.24\}$.

Before showing the results, in Fig. 5.3.1 we see the effect of dispersion over a random geometric graph. It may help to better understand the purpose of applying the dispersion. Using dispersion gives advantages or drawbacks depending, as we can see in Fig. 5.24 and Fig. 5.25, on the chosen separation distance d. The method used for



(a) *Before dispersion*



(b) *After dispersion*

Figure 5.23: *Graph dispersion*

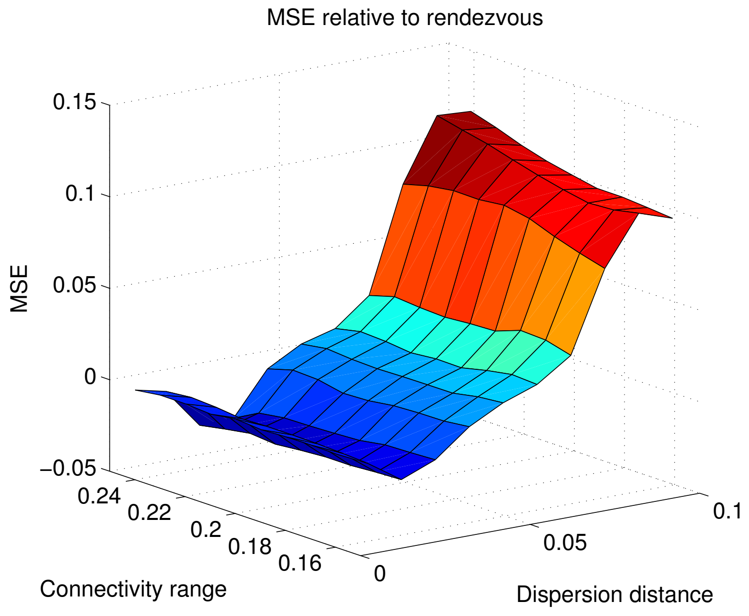


Figure 5.24: *Mean-squared error compared to RV*

comparing with the raw RV is subtracting the MSE value and the number of graph elements obtained for the raw RV method to all the other sets of results with different dispersion distances, thus if the result is negative the dispersion helped otherwise was counter-productive. In both figures we can see the same tendency, when the distance chosen is to high we have a negative maximum because the nodes are spread too far from each other. On the other hand, if we choose d too small the problem is that the dispersion creates many different swarms disconnected. Therefore we have a trade-off and d must be selected taking into account these two constraints.

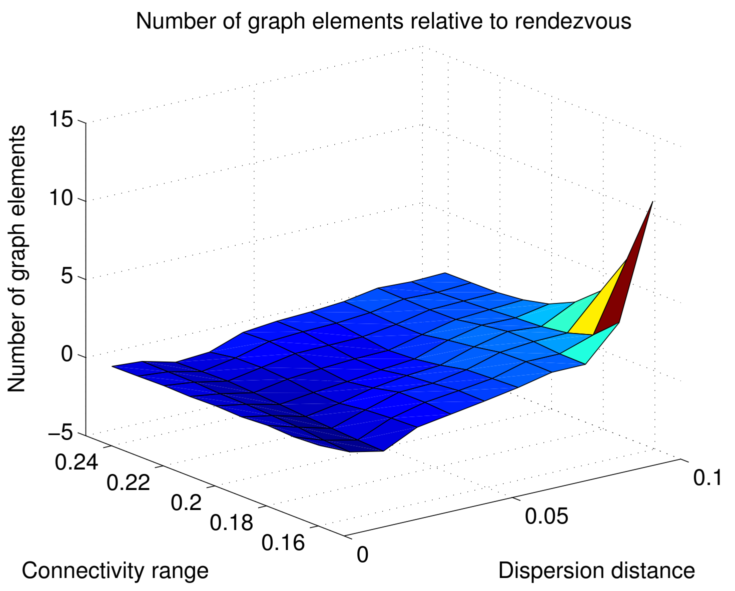


Figure 5.25: *Number of graph elements compared to RV*

5.4 Regular Matrices

To conclude our study of the consensus algorithms we are going to analyze the behavior over regular networks. In particular, in 1D space and over grids for the 2D case. In this section the definition of the connectivity range is changed, now we define it as the initial distance among nodes times a constant $c \in [1, 3]$. We will use 2 different sets of parameters, one for 1D and other for 2D. Considering that we use regular scenarios, it is not necessary to average because there is no need to counteract the effect of any random assets.

In this whole section we will focus on if the network stays connected, therefore we will plot the number graph elements after a number of iterations sufficient to get consensus. As explained in Section 2.2.4, having more than one graph element means that the network has suffered from disconnection.

Parameter	Value
I	4-100
$k_{\max}$	200
r	$c \times d$

Table 5.4: Parameters Section 5.4.1

5.4.1 One Dimensional Scenarios

The algorithms used for this special case are RV and line weights. As explained in Section 3.5.5 the line method is an approach specially designed for the one dimensional environment. We developed it because for 1D scenarios the AW presents issues since the angles are always 0 or 180 grades.

We will plot the number of graph elements depending on different settings. The values for the settings we will use are in Table 5.4.

In Fig. 5.26 we have the results for RV over a 1D scenario. We see how when we have a large number of agents and a small connectivity range the connectivity of the network is affected, ant thus the number of graph elements is bigger. It is possible to appreciate how the number of agents affects more than the connectivity range for RV in 1D.

Now let us focus on the line weights we specifically designed for 1D. Line weights outperform RV since the graph elements are 1 for every setting. This can be seen in Fig. 5.27.

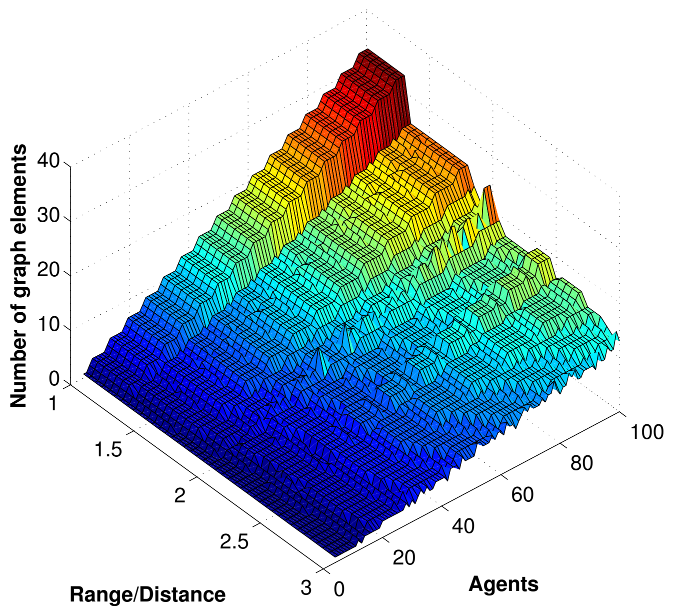


Figure 5.26: *Rendezvous weights*

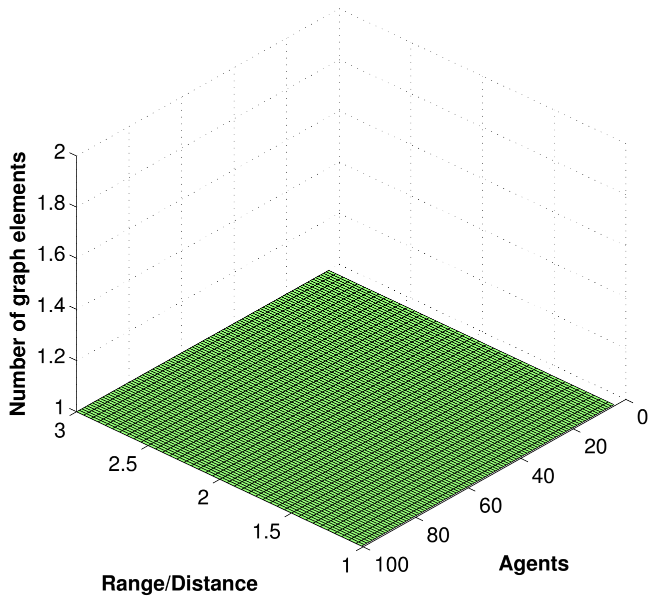


Figure 5.27: *Line weights*

Parameter	Value
I	50-100
$k_{\max}$	100
r	$c \times d$

Table 5.5: Parameters Section 5.4.2

5.4.2 Two Dimensional Scenarios

We test in this section RV and AW, since in previous sections they have proved to be the most effective algorithms for reaching consensus. We seek in this section, as in the section before, whether regular graphs stay connected or not under certain settings. The settings used are in Table 5.5.

Now the results for the RV algorithm are shown in Fig. 5.28. When we have a low number of agents and a high connectivity range we achieve the best results, however what is indeed interesting is that the connectivity range affects more than the number of nodes. Only with high connectivity ranges is possible to reach the objective of having one graph element. This contrast with the 1D case where the number of agents affects more than the connectivity range.

The results for the AW show a clear result, for regular networks as long as the initial graph is connected consensus is always achieved. In Fig. 5.29 is possible to appreciate how for all the values of range and number of nodes the number of graph elements is always one.

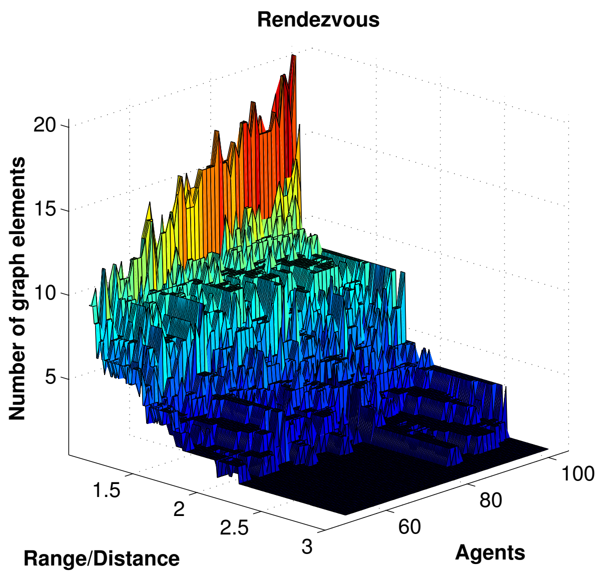


Figure 5.28: *Number of graph elements depending on the connectivity range and the number of nodes*

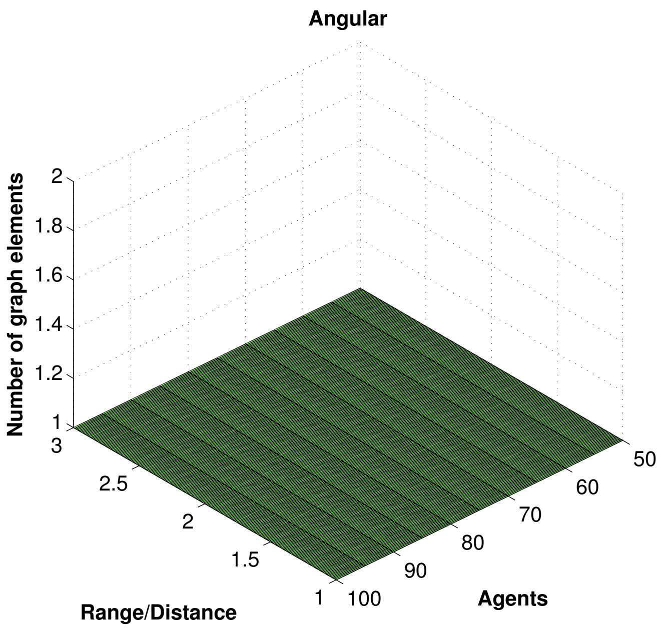


Figure 5.29: *Number of graph elements depending on the connectivity range and the number of nodes*

6 Conclusions and Outlook

This thesis is meant to be an analysis of distributed consensus algorithm applied over multi-agent networks. The objective was studying the rendezvous problem, but other distributed applications such the network formation control were dealt with.

Since the networks can be defined as graphs, it was necessary to revise related theory and also some network topology and matrix theory aspects.

Once the theory basis were settled, we moved into the consensus algorithm design. First we declare the parameters, objectives and assumptions under which we would work for then discussing three weight matrix designs from other scientific works. After noticing their weak points we tried to develop two new weight matrices which would improve the convergence and successful convergence rate of the consensus algorithm.

The next step was introducing some control capabilities in our programs for studying how they would affect the overall performance of the system. In this moment was necessary to create a new consensus algorithm with different conditions in order to adapt it to the control formation problem.

After presenting the main theoretical and design aspects of the consensus algorithm we started showing some of the results obtained during our research. All the four weight designs were compared, for then showing the improvements obtained by applying modifications to the RV algorithm. To conclude that section we brought some of the results obtained for regular networks both in 1D and 2D.

In the future a more detailed theoretical analysis of the consensus algorithm under different conditions can be carried out for explaining the behavior of the different solutions here presented. A good theoretical analysis could therefore lead to the development of better designs which would face the issues we found in this work. A deeper analysis on the formation control and dispersion is of a great interest, not only for consensus purposes, but also for exploration or monitoring applications due to the great possibilities these methods give for generating a controlled enlargement of the networks.

Bibliography

[1] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, pp. 71–82, Apr. 2007.

[2] B. Johanssona, A. Speranzonb, M. Johanssona, and K. H. Johansson, "Technical communique on decentralized negotiation of optimal consensus," *Automatica*, vol. 44, pp. 1175–1179, Dec. 2007.

[3] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[4] C. H. Caicedo-Núñez and M. Žefran, "Consensus-based rendezvous," pp. 1031–1036, IEEE, Sept. 2008.

[5] V. Schwarz, G. Hannak, and G. Matz, "On the convergence of average consensus with generalized metropolis-hastings weights," *Vienna University of Technology, Institute of Telecommunications*, 2014.

[6] B. Bollobás, *Random graphs.* Cambridge Univ. Press, 2nd ed., 2001.

[7] M. Penrose, *Random geometric graphs.* Oxford Univ. Press, 2003.

[8] F. Chung, "Lectures on spectral graph theory," 1996.

[9] R. A. Horn and C. R. Johnson, *Matrix analysis.* (UK): Cambridge Univ. Press, 1999.

[10] D. Cvetković, "New theorems for signless laplacian eigenvalues," vol. 137, pp. 131–146, 2008.

[11] B. Mohar, "The laplacian spectrum of graphs," *Graph Theory, Combinatorics, and Applications*, pp. 871–898, 1991.

[12] R. Merris, "Laplacian matrices of a graph: A survey," *Linear Algebra its Appl.*, vol. 197, pp. 143–176, 1994.

[13] M. Haenggi, J. G. Andrews, F. Baccelli, O. Dousse, and M. Franceschetti, "Stochastic geometry and random graphs for the analysis and design of wireless networks," *IEEE J. Sel. Areas Comm.*, vol. 27, pp. 1029–1046, 2011.

[14] V. Schwarz, *Distributed averaging in wireless sensor networks.* PhD thesis, 2014.

[15] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem," *IEEE Conf. Decision Control*, pp. 1508–1513, 2003.

[16] V. Schwarz and G. Matz, "Average consensus in wireless sensor networks: Will it blend?," pp. 4584–4588, 2013.

[17] V. Schwarz and G. Matz, "Non linear average consensus based on weight morphing," pp. 3129–3132, 2012.

[18] V. Schwarz and G. Matz, "On the performance of average consensus in mobile wireless sensor networks," pp. 175–179, 2013.

[19] R. Freeman, P. Yang, and K. Lynch, "Stability and convergence properties of dynamic average consensus estimators," pp. 338–343, 2006.

[20] L. Xiao, S. Boyd, and S. Lall, "Distribute average consensus with time-varying metropolis weights," 2006.

[21] E. Deza and M. M. Deza, *Encyclopedia of distances.* Springer, 2009.

[22] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automat. Control*, vol. 49, pp. 1465–1467, 2004.

[23] T. Eren, P. N. Belhumeu, and A. S. Morse, "Closing ranks in vehicle formations based on rigidity," pp. 1–6, Dec. 2002.

[24] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," pp. 1–35, Apr. 2003.

[25] W. Ren, "Consensus based formation control strategies for multi-vehicle systems," pp. 4237–4242, June 2006.

[26] J. R. Lawton, R. W. Beard, and B. Young, "A decentralized approach to formation maneuvers," *IEEE Trans. Robot Automat.*, vol. 19, pp. 933–941, 2003.

[27] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automat. Control*, vol. 51, pp. 401–420, Mar. 2006.