Multiagent System Optimization

José Miguel Filipe Antunes

Thesis to obtain the Master of Science Degree in

Engineering Physics

Supervisors: Prof. Doctor Joao Manuel de Freitas Xavier
Prof. Doctor Rui Manuel Agostinho Dilão

May 2016

Hardships often prepare ordinary people for an extraordinary destiny.

C.S. Lewis

Acknowledgements

Esta tese assinala o fim de um ciclo, MEFT. Foi um ciclo cheio de trabalho, cheio de bons momentos, e também momentos duros e de dúvida. Se estou a terminar este ciclo é porque fui apoiado e motivado pelas pessoas que me rodeiam. Por isso, quero deixar por escrito um agradecimento enorme às pessoas que me acompanharam neste caminho.

Ao meu Pai, à minha Mãe e irmãos, António, Nuno e Pedro, por toda a dinâmica que temos juntos, todos os ensinamentos e todo o apoio. Por todos os puxões de orelhas, que me transformaram numa melhor pessoa.

À Inês que me motivou a ir mais além, e a entrar no desconhecido. Por me ouvir, e por repetir n vezes a mesma coisa. Por estar ao meu lado com a força que às vezes me faltou. Pelo amor que partilhamos.

Ao Professor João Xavier, pela sua dedicação e por ser um excelente professor que me mostrou uma nova área do conhecimento. Optimização e Algoritmos foi a cadeira que me impulsionou para este trabalho, pois foi a disciplina que mais gostei de estudar no IST! Agradeço também os métodos de trabalho e as dicas que me foi dando ao longo deste percurso, que me tornaram um melhor estudante. Só assim se consegue trabalhar sem desmotivar e sem ficar preocupado com os problemas que aparecem. Adorei cada momento deste trabalho. Fico feliz por ter contrariado o que sempre me haviam dito, que mais cedo ou mais tarde me iria fartar da tese. Tivesse eu mais tempo, mais coisas iríamos descobrir!

Aos meus novos amigos! Vasco, Beatriz, Sabina, Susana, Zita! Por estarem presentes e pelos almoços e as discussões apaixonadas nos tempos mortos no laboratório! Todos aprendemos uns com os outros e adorei os tempos que passei com vocês (e as novas experiências gastronómicas como o Ad Lib e o Thai Square Lisbon)!

Aos meus camaradas da luta! João Martins, Rúben, Cunha, Natália, Prata, Ana Silva, Tiago Coutinho, Rita, Sofia, Inês, Vasco Pesquita, por fazerem do IST um local com mais graça! Pelos inúmeros cafés, discussões e piadas! Testes, exames, alegrias e frustrações que partilhámos! E não esquecendo nunca, Simon e Anastasiia, Diana, Ana Neves, Ana Silva e Zé Amaral, pelo que me ensinaram e pelo que cresci com vocês!

E por fim, mas não menos importante, os amigos que cresceram comigo. Ricardo, Pedro, Dani, Maria. Porque um pouco do que eu sou hoje vem de todos os tempos que passámos juntos. Obrigado.

Resumo Analítico

O consenso é uma ferramenta importante em processamento distribuído. O objectivo dos algoritmos de consenso é calcular a média das medidas de cada agente da rede. À medida que estes trocam informação, a informação que guardam aproxima-se do valor da média.

Algoritmos de consenso têm duas vertentes: canais de comunicação e velocidade. Devem chegar rápido ao resultado, usando o mínimo de canais.

Neste trabalho, desenvolvemos algoritmos que melhoram o estado da arte nestas duas vertentes, separadamente.

Na primeira parte, desenhámos um método que reduz o número de canais de comunicação tendo um impacto pequeno na velocidade de convergência. Através de técnicas de optimização convexa e da teoria da dualidade, reduziram-se o número de canais em cerca de 80%, sofrendo apenas um impacto de 5% na performance da rede. Ou seja, encontrou-se um conjunto de canais que mantêm um rápido processamento da informação.

Na segunda parte, desenhámos métodos para acelerar a velocidade de convergência. Para redes assimétricas (onde a comunicação bidireccional não é garantida), este é um problema difícil de resolver, pois o raio espectral é uma função não convexa. Propusemos uma série de aproximações convexas que aumentam a velocidade de algoritmos de consenso em cerca de 45%, para o mesmo número de canais de comunicação.

Keywords: Consenso, algoritmos distribuídos, redes esparsas, optimização convexa, minimização do raio espectral

Abstract

In distributed multi-agent networks, consensus is a core tool. The goal of any consensus algorithm is to compute the average of the measurements that each agent holds. As they exchange information the average should be reached at each agent. Consensus algorithms use two key resources: communications and time. They thrive to be fast while using a modicum of communication channels.

We develop consensus algorithms that substantially improve the state-of-art in these two directions, separately.

In the first part, we design methods that reduce the number of communication channels needed, while having a negligible impact on the speed of the consensus algorithm. By exploring tools from convex optimization and duality theory, we are able to trim the number of channels up to 80%, while paying only about a 5% penalty in convergence speed. In other words, for a given network topology, our methods spot a subset of the available channels that still sustain a fast processing of information.

In the second part, we develop methods that accelerate the speed of convergence. We address this problem for the challenging setup of asymmetric networks, i.e., networks in which agents may not communicate bidirectionally. This task translates into the minimization of the non-convex spectral radius of asymmetric matrices whose entries are constrained to a convex set. We propose several reformulations of this non-convex problem that lead to efficient convex approximations and boost the speed of existing consensus algorithms by 45% with the same number of channels.

Keywords: Consensus, distributed algorithms, sparse networks, convex optimization, spectral radius minimization

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 The consensus problem

Consider a set of n agents linked by a communication network. A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ serves as an abstract representation of the network where $\mathcal{V} = \{1, \ldots, n\}$ is the set of agents and $\mathcal{E}$ is the set of edges. An edge $\{i, j\} \in \mathcal{E}$ represents a physical channel between agents i and j.

Agent i holds the number $x_i \in \mathbf{R}$, and the goal of the consensus problem is to compute the average

$$\overline{x} := \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1.1}$$

in a distributed manner by means of an algorithm that only exchanges messages through the edges of the graph, i.e., each agent i can only interact with its neighbours, $N_i := \{j \,|\, \{i, j\} \in \mathcal{E}\}$. The algorithm must produce the average $\overline{x}$ at all agents asymptotically, as the number of exchanged messages grows to infinity.

Distributed versus centralized processing. The consensus problem is trivial in a centralized setting: if there exists a central node, connected to all the nodes of the graph, then the central node can receive all the network data $\{x_i\}$, compute $\overline{x}$, and broadcast it to all nodes. This thesis considers the more challenging and realistic scenario of communication networks without central nodes—see Figure 1.1. When central nodes are absent, the computation of the average is performed in a distributed manner, by iteratively exchanging information between neighboring sensors.

J. Tsitsiklis developed the first distributed methods in 1984 [1]. In his PhD, he studied how some real life problems cannot be solved by a single decision maker (man or machine) as they are too large for this classical method. In distributed processing, it is possible to mitigate some of the classic methods' problems by breaking a big task into smaller ones, which are addressed by several nodes. This imposes new problems such as inter-node communication. There are different methods for node communication, one being a gossip type communication, where each node talks with a single neighbour at random times, while another is the method this work relies on—the synchronous communication of each node with all its neighbours.

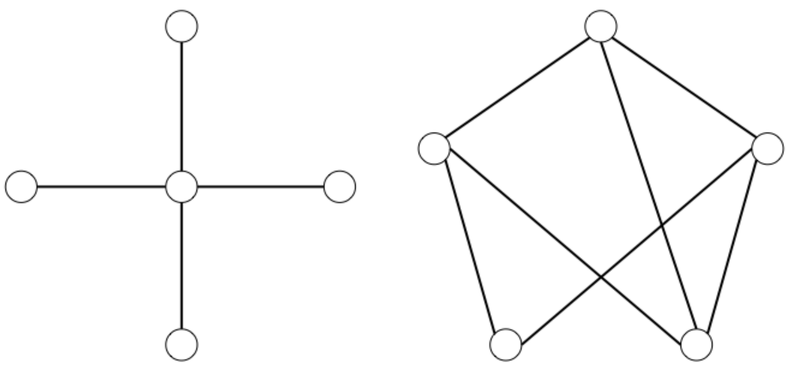


Figure 1.1: Left: network with a central node. Right: network without a central node.

One advantage of distributed methods is scaling. As a network (which relies on a couple of central nodes) increases in size, communication constraints such as bottlenecks due to high data volumes become noticeable and worsen the network's performance. Also, central nodes, like any other node, are subject to failure. When a central node fails the system fails, since they are the nodes that support the whole network by performing calculations and broadcasting results.

Distributed linear iterations. To solve the consensus problem, agents cooperate via an iterative process that exchanges messages through the edges of the graph. Let $x_i^{(t)}$ be the estimate of $\overline{x}$ available at agent i at time $t = 0, 1, 2, \ldots$ This estimate is known as the state of the agent.

A popular algorithm based on linear iterations can be described as follows: the algorithm is initialized as $x_i^{(0)} := x_i$ for $i = 1, \ldots, n$, and the states are updated as:

$$x_i^{(t+1)} = W_{ii} x_i^{(t)} + \sum_{j \in N_i} W_{ij} x_j^{(t)} \quad , i = 1, \ldots, n. \tag{1.2}$$

Equation (1.2) shows that each agent repeatedly performs a weighted average of its estimate with its neighbours' estimates. Equation (1.2) can be rewritten in a compact vector notation:

$$x^{(t+1)} = W x^{(t)} \tag{1.3}$$

where $x^{(t)} := (x_1^{(t)}, \ldots, x_n^{(t)}) \in \mathbf{R}^n$ is called the network state, and $W = (W_{ij}) \in \mathbf{R}^{n \times n}$ is the weight matrix. Note that $W_{ij} = 0$ whenever $\{i, j\} \notin \mathcal{E}$, i.e., the non-zero entries of W encode the graph topology.

The design of W is a critical issue as not all weight matrices lead to consensus, that is, not all matrices W ensure that

$$x_i^{(t)} \to \overline{x} \quad \text{as} \quad t \to \infty \quad \text{,for all} \quad i = 1, \ldots, n. \tag{1.4}$$

Chapter 2 reviews the necessary and sufficient conditions of W that secure the desired asymptotic behaviour in equation (1.4).

Goal. In typical multi-agent systems, each node has limited computational, communication, and energy resources. Therefore, it is of the utmost importance to have a fast convergence in equation (1.4) with minimal resource expenditure.

In the scope of this work, the cost consists in the amount of communication channels used and the amount of iterations performed. Thus, our aim is to reduce the cost of computing the average in a distributed manner without severely hindering the speed of convergence.

1.2 Applications of consensus

This section showcases the real-life application of distributed consensus in illustrative examples from statistical signal processing, computer science, and robotics/control.

Temperature sensing. This application is usually given as a *toy example* [2, 3], to introduce the functionality of multi-agent systems as it is simple to understand and interpret.

In a big set of temperature sensing devices, each device will communicate with the sensors around it, and together form a network spanning a considerable area. When such a network is considered, the question "what is the temperature at node X?" is not as relevant as "what is the average temperature measured?" or "what areas are above Y degrees Celsius?", due to the general state of the network being more important than a single node's, as nodes can malfunction and deliver wrong readings [3, 4].

The question "What is the average temperature measured?" is hard to answer if a central node does not exist. The simple act of sending all the information to a central entity and then broadcasting a result no longer applies in this scenario, and the distributed consensus problem comes into action.

Disaster relief. In the instance of a natural disaster, such as wildfire, thousands of disposable sensors can be densely scattered over a disaster area [5]. Some sensors might be destroyed but the remaining will be able to create a network that captures information, maps the area to deliver evacuation routes, and helps emergency response teams.

The working principle of this application is the same as that of terrain or habitat monitoring, where the deployed sensors form a network, and capture environmental data due to the terrain's inaccessibility [3, 5, 6].

Vehicle tracking. The robotics/control research area also has use for multi-agent systems. Each robot can be considered a node in the network and achieving a consensus here would mean, for example, finding a position all robots agree to use as a meeting point.

As specified by Fax and Murray [7], extensive work has been done in this area specifically in microsatellite clusters, unmanned aerial vehicles (UAV's), autonomous underwater vehicles (AUV's), and automated highway systems. While these areas pose different challenges, they all have in common their approach on how to control vehicle dynamics with the two main methodologies being "leader-follower" and "virtual leader".

In "leader-follower", all vehicles have their trajectories clearly defined by a leader with whom they communicate. However this strategy is undesired since over-reliance on a single machine can be problematic, especially in adversarial environments, where damage to the leader machine endangers all others.

The “virtual leader” method relies on communication between machines to reach a consensus and create a “virtual leader” whose trajectory is then communicated and followed by everyone. The downside of this approach is the high communication and computation capabilities needed to make it work.

Network Load. It is also possible to control the load on a set of processors [8] by using equation (1.3).

The case where the weight matrix W is equal to $I - \alpha L$ allows for a visualisation of this control — I is the $n \times n$ identity matrix, L is the graph's Laplacian matrix and α is a known parameter. The Laplacian matrix of a graph holds a node's number of connections in the diagonal and the value -1 in the entry (i, j) if $(i, j) \in N_i$.

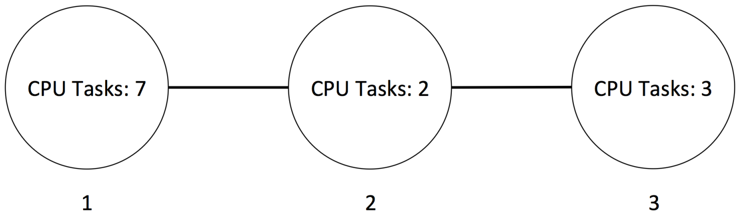


Figure 1.2: Load in a network of CPU's.

Consider the network in figure 1.2. Its corresponding Laplacian and Weight matrix are given by:

$$L = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \Rightarrow W = \begin{bmatrix} 1-1\alpha & 1\alpha & 0 \\ 1\alpha & 1-2\alpha & 1\alpha \\ 0 & 1\alpha & 1-1\alpha \end{bmatrix}. \tag{1.5}$$

Given equation (1.3), it is possible to calculate how a certain node will be affected in time step $(t+1)$. Take for example node 2:

$$\begin{aligned} x_2^{(t+1)} &= \alpha x_1^{(t)} + (1-2\alpha)x_2^{(t)} + \alpha x_3^{(t)} \\ &= x_2^{(t)} + \alpha(x_1^{(t)} - x_2^{(t)}) + \alpha(x_3^{(t)} - x_2^{(t)}) \end{aligned} \tag{1.6}$$

It is possible to observe from equation (1.6) that the nodes with lower CPU tasks will receive tasks from nodes with a bigger workload. This way, the computer network balances its workload through each iteration of the algorithm, leading to an even load balance.

Hypothesis Testing. Another interesting field where distributed methods are important and have already been explored is Hypothesis Testing [9]. Consider a scenario where two boxes emit random numbers at a constant frequency. These two boxes are placed inside a black box, and, as such, it is impossible for the user to see which box is sending the numbers (figure 1.3).

The user does have a model for boxes H1 and H2. It is up to the user to collect data emitted from the black box and test it against existing models. This shows a simple hypothesis testing, where each box represents an unknown state of nature (fire/no fire; target/no target; etc.), and the observations are used to infer which hypothesis is active.

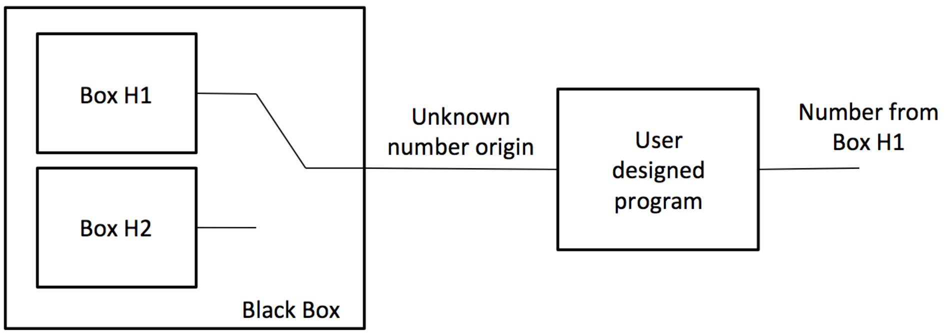


Figure 1.3: Black box that emits random numbers.

Assuming box H1 and H2 have two associated probability density functions, $p_{H1}(x)$ and $p_{H2}(x)$, as soon as the first number x_1 is emitted, will it be possible to check its origin?

As the user has models for each box, the user can calculate the probability $p_{H1}(x_1)$ and $p_{H2}(x_1)$, and choose the model with the highest probability score, *i.e.*, the more plausible one.

Considering each observed value is independent from previously emitted ones, this formulation can be changed into a ratio and generalized for n observations with the geometric mean:

$$\frac{p_{H1}(x_1)}{p_{H2}(x_1)} \underset{H2}{\overset{H1}{\gtrless}} 1 \Rightarrow \left(\frac{p_{H1}(x_1)}{p_{H2}(x_1)} \cdots \frac{p_{H1}(x_n)}{p_{H2}(x_n)}\right)^{1/n} \underset{H2}{\overset{H1}{\gtrless}} 1 \tag{1.7}$$

By applying the logarithm it is possible to modify equation (1.7) to fit a distributed linear averaging method, equation (1.8).

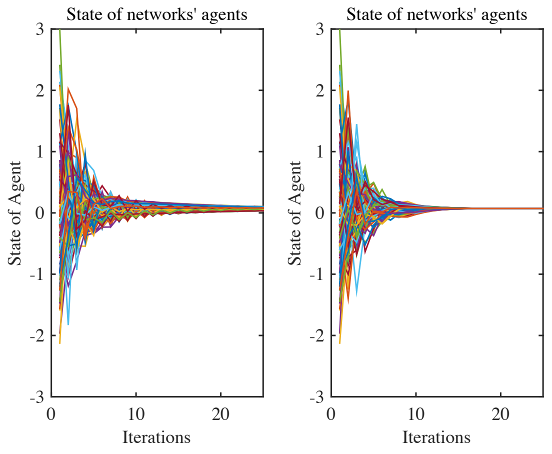
$$\frac{1}{n}\log\left(\frac{p_{H1}(x_1)}{p_{H2}(x_1)} \cdots \frac{p_{H1}(x_n)}{p_{H2}(x_n)}\right) = \frac{1}{n}\left[\log\left(\frac{p_{H1}(x_1)}{p_{H2}(x_1)}\right) + \cdots + \log\left(\frac{p_{H1}(x_n)}{p_{H2}(x_n)}\right)\right] \underset{H2}{\overset{H1}{\gtrless}} 0 \tag{1.8}$$

This new formulation of the problem, (1.8), allows for a sensor's network to be installed, where agent i measures x_i, performs a local computation (computing the logarithm of the ratio of probabilities), and engages with neighbours to compute the average. As the information is exchanged and the average calculated, the final result will allow to deduce which statistical model underlies the observations.

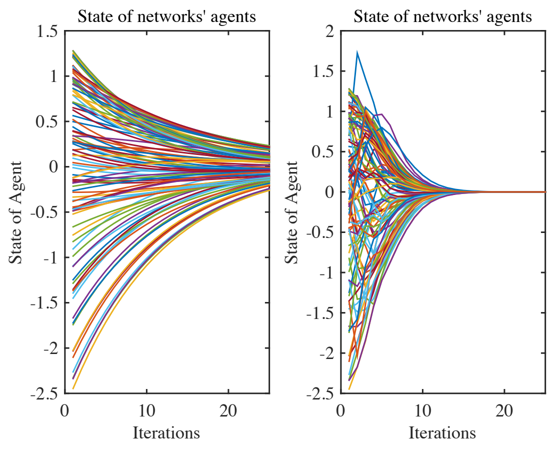
1.3 Impact of thesis: three scenarios

As an illustration of the results achieved in this thesis, examples of the significant speed-up achieved in three applications are given. The state of the art method (which is called the naive method) is compared with this thesis' best one, the Cool Down method, which is developed in detail in section 3.5.

Standard consensus. The standard consensus is what all weight matrices obtained from the methods in this work must be able to reach. This means that, given a starting state of the network, all agents must converge to the average of the initial state. This idea can be applied in several different cases that will be shown in the following examples. This example serves as a statement to the results obtained in this work. Observe figures 1.4a and 1.4b. Figure 1.4a compares how the state of the agents progress through each communication iteration starting from a random vector. It is possible to observe that the network optimized via the Cool Down method obtains consensus in a much smaller number of iterations. However, to make this explicit, look at figure 1.4b. In this figure, the starting vector was purposely chosen to be the worst possible vector to initialize the agents' states. It is even clearer that the optimized network has reached a consensus at 15 iterations, and the naive method is still far away from a consensus at 25 iterations.



(a) Naive vs Cool Down optimization for a random vector.



(b) Naive vs Cool Down optimization for a worst case vector.

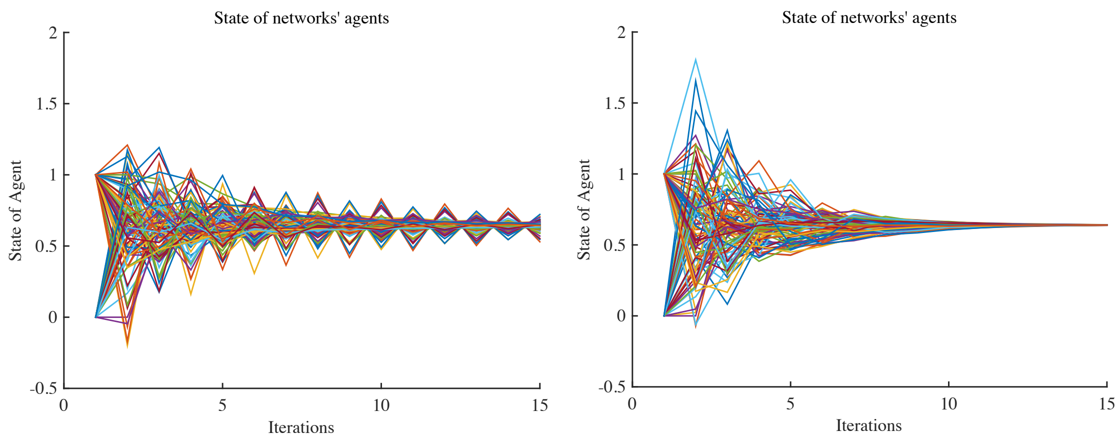
Figure 1.4: State of agents as they communicate to reach consensus.

Event detection. This application concerns hypothesis testing. Imagine there is a set of sensors that measure a physical event and output a 1 or a 0 depending if the event is happening or not. Some sensors may malfunction, but in general knowing if the event is happening is the same as knowing the state of the network, which is given by the average of the outputs.

As with the previous examples, the network that is better optimized will reach a consensus faster, thus providing relevant information first. Look at figures 1.5a and 1.5b. The images refer to the values each node assumes after communicating with its neighbours. It is possible to observe that, even after 10 inter-agent communications, the status of the network isn't well defined, and oscillates a lot around the consensus value for some agents. On the other hand, for the well optimized network, all agents have a very similar opinion at iteration 10, thus making it possible to know if the event has happened with more certainty than the naively optimized network.

Offline Target tracking. Imagine a situation where a network of distance measuring sensors has the task of tracking an object through time. Each sensor has a noise level and only by cooperating can they achieve an answer that is suitable to the user.

The setting of the problem is the following: an object moves and it's position is captured (with an error) at regular intervals. Between each capture, the sensors can exchange information to improve their estimates but the amount of times they can communicate is limited. The objective is to give to the



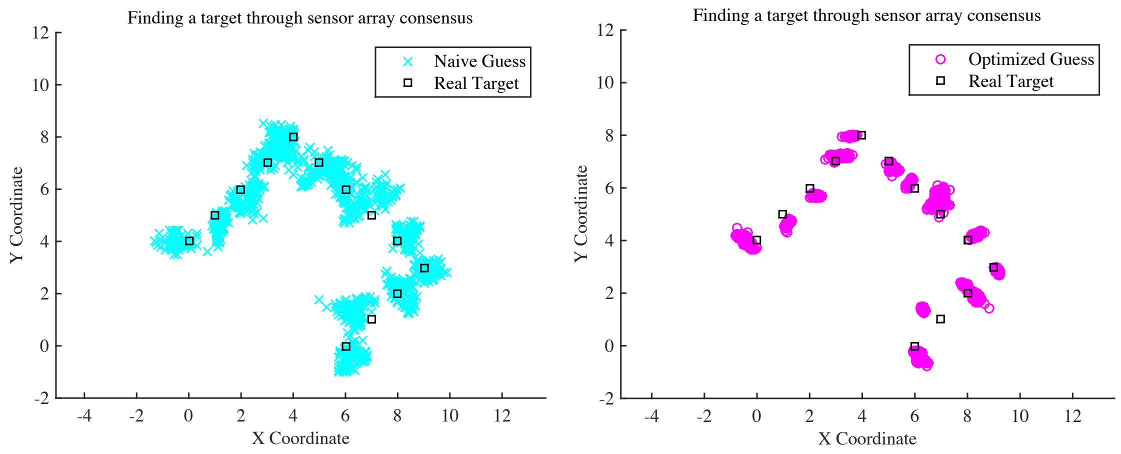
(a) Naive optimization.

(b) Cool Down method.

Figure 1.5: State of agents as they communicate to reach consensus.

user the ability to pinpoint with some accuracy the path that was traveled by the object.

The following images compare the results obtained when a network of sensors tries to reach a consensus with a naively optimized network (1.6a), and the results from a network that was optimized through the best method found by the work done in this thesis (1.6b).



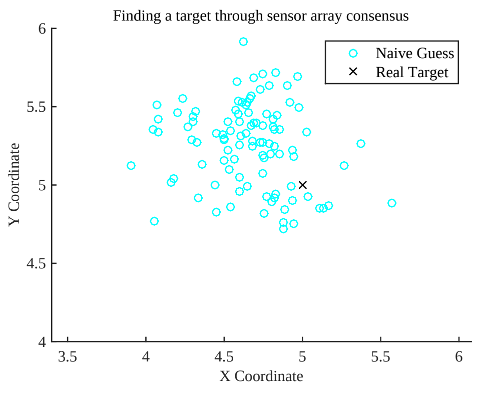
(a) Naive optimization.

(b) Cool Down method optimization.

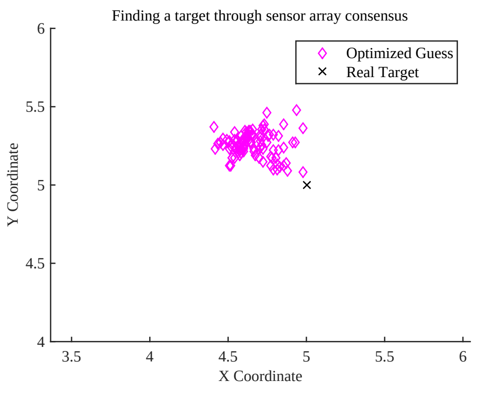
Figure 1.6: Tracking a target through a sensor's network with different optimizations.

Online Target tracking. In online target tracking, the sensors make an observation, and estimate it against previous observations they have made. Once the most recent estimate is complete, they exchange information with their neighbouring sensors, again with a limited amount of information transactions. The network that has the best performance when trying to reach a consensus with its neighbours will be closer to the real target as the successive observations are made.

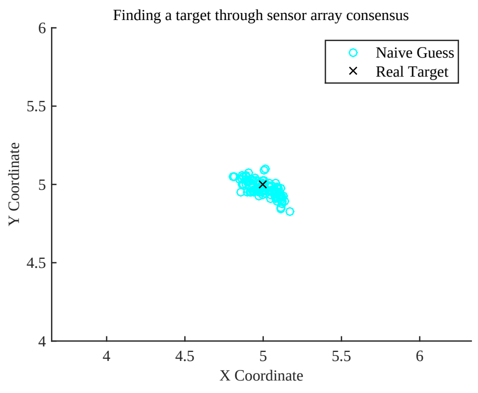
The following images are snapshots of the estimates of each sensor for their observations at different times. Figures 1.7a and 1.7b are snapshots of the first observation (it makes no estimation because there are no previous observations), figures 1.7c and 1.7d are snapshots of the fifth timeframe, and figures 1.7e and 1.7f are snapshots at the tenth timeframe.



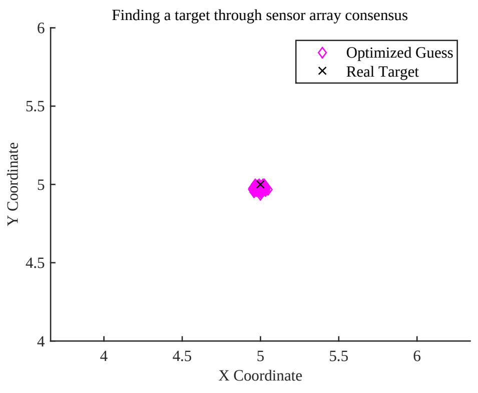
(a) Naive optimization at iteration 1.



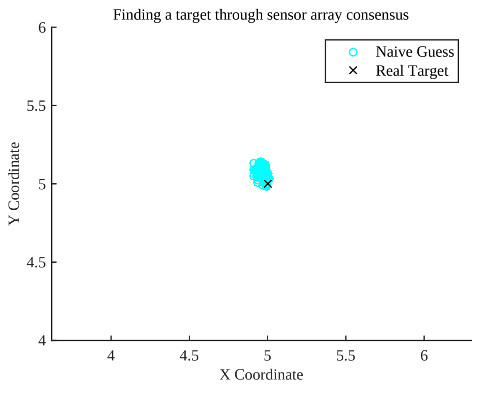
(b) Cool Down optimization at iteration 1.



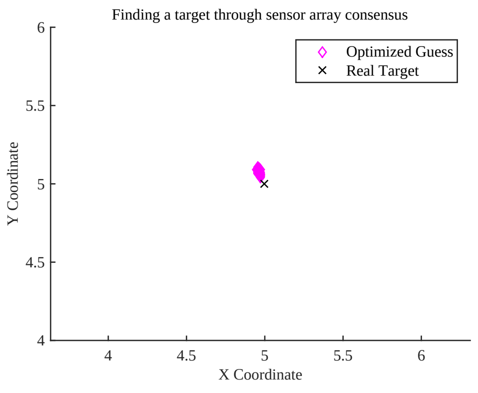
(c) Naive optimization at iteration 5.



(d) Cool Down method at iteration 5.



(e) Naive optimization at iteration 10.



(f) Cool Down method at iteration 10.

Figure 1.7: Online target tracking through a sensor's network with different optimizations.

Ultimately, it is possible to observe that figures 1.7d and 1.7f are better estimates of the real target, as the spread the data points have is tighter than the ones in figures 1.7c and 1.7e. This happens because the optimized network reaches a consensus faster than the naive network. As such, the information exchanged at each iteration is closer to reality allowing better estimates with the data obtained.

Note: After the thesis was defended, the authors learned that reference [10] proposed an almost identical algorithm to the Cool Down algorithm that was developed in Section 3.5.

1.4 Convex Optimization

Though there are many tools and ideas to approach the optimal design of matrix W in equation (1.3), this thesis focuses on using Convex Optimization to achieve its results.

Optimization problems exist everywhere in the real world, from economics to electronics and sensor systems. A simple way to solve an optimization problem is to embed in each of these systems a Convex Formulation of the problem, and design an algorithm to find the answers in real time. The standard form a Convex Formulation has is the following:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, ..., m \\ & h_i(x) = 0, i = 1, ..., p. \end{aligned} \tag{1.9}$$

Looking at equation (1.9), $f_0(x)$ is the objective (or cost) function and $g_i(x)$ and $h_i(x)$ the constraint functions. In a generic optimization problem the result is obtained by minimizing the incurred cost, while respecting the constraints that limit the solution's search area.

The value that solves the problem and respects all constraints, x^*, is called the optimal value, and gives the lowest value of $f_0(x)$.

Equation (1.9) serves as a framework where different types of optimization problems fit. The most generic and versatile class formulation is called Semidefinite Programing (SDP) which, with progressive simplifications, can be specialised into other classes like Second Order Cone Programming (SOCP), Quadratic Programming (QP) and Linear Programming (LP).

As this thesis focuses on the application of the SDP formulation, examples of the remaining types of optimization problems will be left out as these are thoroughly explained in Boyd's "Convex Optimization" [11].

Semidefinite Programming. An SDP is a conic form problem where the cone K contains the positive semidefinite $k \times k$ matrices, $\mathcal{S}^k_+$. Equation (1.9) assumes the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & x_1 F_1 + ... + x_n F_n + G \underset{K}{\preceq} 0 \\ & Ax - b = 0, \end{aligned} \tag{1.10}$$

where $G, F_1, ..., F_n \in \mathcal{S}^k$, and $A \in \mathbf{R}^{p \times n}$.

In this type of problems, constraints are matrix inequalities. This is the most generic class of problems, and although all other convex optimization problems can be inflated in complexity to fit this framework, it is wise not to do so. The techniques used to formulate the problems presented in this thesis into an SDP form will be explained in the chapters where they are applied.

Chapter 2

Weight matrix design

2.1 Distributed Linear Averaging conditions

2.1.1 Weight matrix constraints

The core problem in consensus algorithms is to design the matrix W. This chapter reviews the conditions on W that lead to consensus [12].

First and foremost, matrix W needs to belong to the group of matrices that can represent the given graph:

$$W \in \mathcal{S} \Rightarrow \{W \in \mathbf{R}^{n\times n} : w_{ij} = 0 \text{ if } \{i,j\} \notin \mathcal{E} \text{ and } i \neq j\}. \tag{2.1}$$

This is an important constraint as it allows the algorithm to be applied in real life systems. If $w_{ij} \neq 0$ if $\{i,j\} \notin \mathcal{E}$, the algorithm would try to communicate through a physical channel that doesn't exist in the communication network. It is possible to pose equation (1.3) in a different form which only relies on the initial values' vector, $x^{(0)}$:

$$\begin{aligned} x^{(1)} &= Wx^{(0)}, \\ x^{(2)} &= Wx^{(1)} = WWx^{(0)}, \\ &\dots \\ x^{(t)} &= W^t x^{(0)}. \end{aligned} \tag{2.2}$$

Secondly, the consensus problem states that when $t \to \infty$, $x^{(t)}$ tends to the average value. The average value, $\bar{x}$, can be represented in the following form:

$$\bar{x}\mathbf{1} = \left(\frac{\mathbf{1}^T x^{(0)}}{n}\right)\mathbf{1} = \left(\frac{\mathbf{1}\mathbf{1}^T}{n}\right)x^{(0)}, \tag{2.3}$$

with n being the number of nodes in the network and $\mathbf{1}$ being the n-sized vector of ones. A simple limit can thus be written, which will give another constraint that matrix W must respect:

$$\lim_{t\to\infty} x^{(t)} = \lim_{t\to\infty} W^t x^{(0)} = \left(\frac{\mathbf{1}\mathbf{1}^T}{n}\right)x^{(0)} \Rightarrow \lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^T}{n} \tag{2.4}$$

And finally, assuming the limit shown in equation (2.4) holds, it is possible to define the *asymptotic* *convergence factor* [12] as,

$$r_{\text{asym}}(W) = \sup_{x^{(0)}\neq\bar{x}} \lim_{t\to\infty} \left(\frac{\left|\left|x^{(t)} - \bar{x}\right|\right|_2}{\left|\left|x^{(0)} - \bar{x}\right|\right|_2} \right)^{1/t} \tag{2.5}$$

This factor indicates how slowly a sequence is converging to a certain value. Given equations (2.1), (2.4) and (2.5) it is now possible to formulate the problem to solve:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & r_{\text{asym}}(W) \\ \text{subject to} \quad & W \in \mathcal{S} \\ & \lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^T}{n} \end{aligned} \tag{2.6}$$

The formulated problem in equation (2.6) aims to minimize the slowest convergence rate, and restricts W to the mentioned constraints.

2.1.2 Weight matrix convergence conditions

It is important to explore the conditions under which the converging matrix W exists as the problem expressed in equation (2.6) will only have a solution in that case.

Firstly it should be noticed that an important property, which W needs to respect, is the *conservation of mass*. Visually, this is the equivalent of saying that if each node has a given quantity of sand, the objective is to spread it equally throughout all nodes without adding or removing sand. This means that the sum, and therefore the average, of all elements in $x^{(t)}$ is conserved after each iteration, as shown by equation (2.7).

$$\begin{aligned} &\mathbf{1}^T x^{(t+1)} = \mathbf{1}^T x^{(t)} \Leftrightarrow \\ &\mathbf{1}^T W x^{(t)} = \mathbf{1}^T x^{(t)} \Leftrightarrow \\ &\mathbf{1}^T W = \mathbf{1}^T. \end{aligned} \tag{2.7}$$

Furthermore, if a vector $x^{(t)}$ of constant entries is input in equation (1.3), then $x^{(t+1)}$ should be equal to $x^{(t)}$. This means that $\mathbf{1}$ is a right eigenvector of W with associated eigenvalue one, and this condition is expressed by:

$$W\mathbf{1} = \mathbf{1}. \tag{2.8}$$

Given (2.7) and (2.8), limit (2.4) occurs only if the absolute value of the eigenvalues of $W - \mathbf{1}\mathbf{1}^T/n$ is smaller than one. Therefore, the final condition that needs to hold is:

$$\rho\left(W - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) < 1 \tag{2.9}$$

where $\rho(\,\cdot\,)$ denotes the spectral radius, which is also indicative of the convergence speed, and therefore, the following equality holds:

$$r_{\text{asym}}(W) = \rho\left(W - \frac{\mathbf{1}\mathbf{1}^T}{n}\right). \tag{2.10}$$

To prove that the assumption made in equation (2.4) really exists given the conditions provided above , it is crucial to demonstrate its mathematical necessity and sufficiency.

Sufficiency. When W respects (2.7) and (2.8) the following equalities apply:

$$\begin{aligned} W^t - \frac{\mathbf{1}\mathbf{1}^T}{n} &\overset{(2.8)}{=} W^t\left(I - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) \\ &= W^t\left(I - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)^t \\ &\overset{(2.7)}{=} \left(W\left(I - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)\right)^t \end{aligned} \tag{2.11}$$

where the second equality is valid because $\left(I - \mathbf{1}\mathbf{1}^T/n\right)$ is a projection matrix ($P^2 = P$). Taking into consideration equation (2.9), the desired convergence in equation (2.4) is achieved,

$$\lim_{t\to\infty} W^t - \frac{\mathbf{1}\mathbf{1}^T}{n} = 0 \Leftrightarrow \lim_{t\to\infty}\left(W - \frac{\mathbf{1}\mathbf{1}^T}{n}\right)^t = 0. \tag{2.12}$$

Necessity. To prove necessity, it is considered that the limit (2.4) exists if, and only if, W can be decomposed through the Jordan canonical form as

$$W = T\begin{bmatrix} I_k & 0 \\ 0 & Z \end{bmatrix} T^{-1}, \tag{2.13}$$

where I_k is the k-dimensional identity matrix ($0 \le k \le n$) and Z is a convergent matrix ($Z^t \to 0$ and $\rho(Z) < 1$).

Considering $u_1, \ldots, u_n$ the columns of T, and $v_1^T, \ldots, v_n^T$ the rows of T^{-1}, limit (2.4) becomes:

$$\begin{aligned} \lim_{t\to\infty} W^t &= \lim_{t\to\infty} T\begin{bmatrix} I_k & 0 \\ 0 & Z^t \end{bmatrix} T^{-1} \\ &= T\begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} T^{-1} \\ &= \sum_{i=1}^{k} u_i v_i^T. \end{aligned} \tag{2.14}$$

With $u_i v_i^T$ being a rank-one matrix, and its sum $\sum_{i=1}^{n} u_i v_i^T = TT^{-1} = I$ having a rank n, then $\sum_{i=1}^{k} u_i v_i^T$ must have rank k. Comparing (2.4) with (2.14) shows $k = 1$ and $u_1 v_1^T = \mathbf{1}\mathbf{1}^T/n$, which implies u_1 and v_1 are multiples of $\mathbf{1}$, *i.e.*, one is a single eigenvalue of W, and $\mathbf{1}$ is its associated left and right eigenvectors (equations (2.7) and (2.8) hold). Additionally, equation (2.9) holds due to:

$$\rho\left(W - \frac{\mathbf{1}\mathbf{1}^T}{n}\right) = \rho\left(T\begin{bmatrix} 0 & 0 \\ 0 & Z \end{bmatrix} T^{-1}\right) = \rho(Z) < 1. \tag{2.15}$$

In conclusion, the conditions on W are: a spectral norm smaller than 1 to allow convergence ($\rho(W - 11^T/n) < 1$), the conservation of mass ($1^T W = 1^T$) and the fixed point condition ($W1 = 1$) which states that a vector of constant entries remains the same after each iteration.

2.2 Simple Convex Problem

Initial Network. In order to solve the consensus problem with a low cost, the number of connections used by a communication network needs to be minimized while good convergence speed is maintained.

To better understand the problem at hand, a randomly generated network was created with the following features:

- Total number of nodes: 100;
- Maximum number of connections between nodes: 20;
- Total network connections: 765;
- Average number of connections per node: 15.

This network was created to test all the optimization ideas and compare results. Figure 2.1 gives a visual representation of the network.

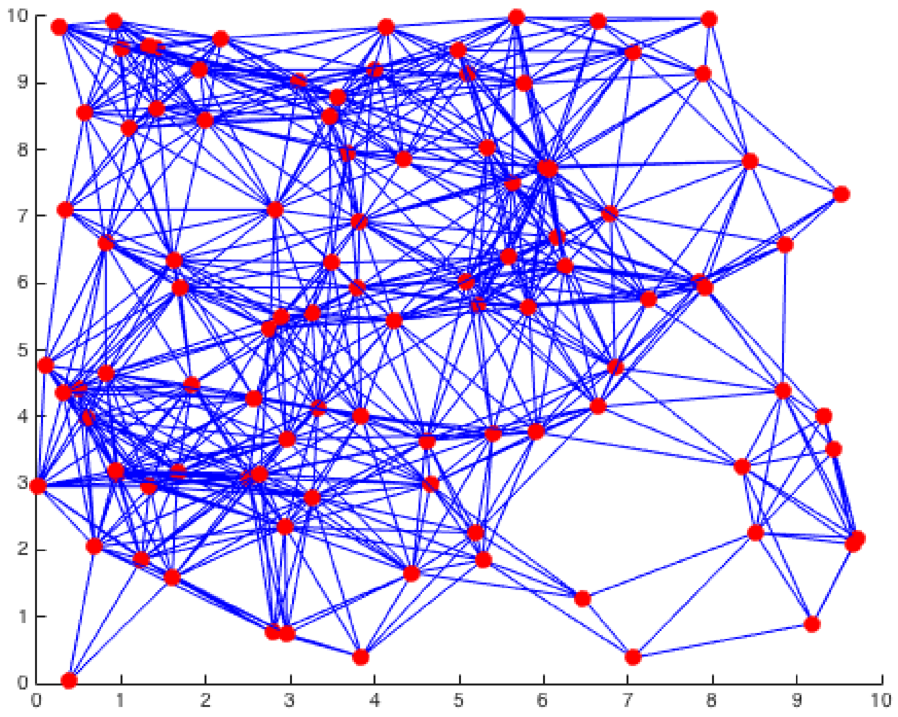


Figure 2.1: Network used as a starting point in the minimization problem.

Equation (2.16) relates the total number of connections between agents and the cardinality (number of non-zero elements) of the matrix that represents the network.

$$\text{\#connections} = \frac{\mathbf{card}(W) - \text{\#agents}}{2} \tag{2.16}$$

Formulation. With the conditions on matrix W defined in 2.7, 2.8 and 2.9, problem (2.6) can now be formulated. The simple case of the Distributed Linear Averaging considers a bidirectional communication network, where weight w_{ij} is equal to w_{ji} in matrix W. Mathematically, this condition is associated with an undirected graph and a symmetric matrix W. When a symmetric matrix is considered, the spectral radius can be simplified to the spectral norm and constraint (2.9) becomes:

$$\left\| W - \frac{11^T}{n} \right\|_2. \tag{2.17}$$

Finally, the problem to solve can be formulated as:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \left\| W - \frac{11^T}{n} \right\|_2 \\ \text{subject to} \quad & W = W^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{aligned} \tag{2.18}$$

The achieved results for W will be hardwired into each node, and nodes will then be able to process the information in their neighbourhood, and approximate the average after each iteration.

Results. The simple optimization problem above shown does not take into account that the objective is to reduce cost through limiting the number of connections used. Therefore, it is only focused on obtaining the fastest convergence speed, which in mathematical terms translates to having the lowest spectral norm. Table 2.1 shows the results obtained from the optimization problem (2.18).

Table 2.1: Cardinality, spectral Norm, and number of connections of matrix W.

Cardinality	Spectral Norm	Connections
1630	0.8451	765

As the following formulations will try to minimize the incurred cost, this spectral norm value is the lower bound, since the fastest convergence speed can only be obtained by using the maximum number of connections possible.

A simple algorithmic approach to the problem of how to find a network that uses less connections, and has the best performance, is to do a greedy brute force search. Algorithm 1 shows the implemented idea when applied n times for the n edges that need to be removed. It is very inefficient because it

Algorithm 1 Greedy brute force search

```
Input a network
for each edge in the network do
    Remove edge;
    Solve FDLA;
    $score(edge) \leftarrow$ spectral norm of $W$;
    Reinstate edge;
end for
Find the $score(edge)$ closest to the lower bound of the spectral norm;
Definitely remove the corresponding edge.
```

solves problem (2.18) for each edge that exists in the network. Thus it needs a very high number of calculations just to determine what edge to remove at each iteration.

The upside to the application of this algorithm is that, if it is applied used until the graph representing the network becomes a tree graph, it establishes a greedy benchmark for all possible number of communication channels in the network.

Figure 2.2 shows the evolution of the spectral norm as the number of edges is reduced. A small increase in spectral norm can be seen in the as the number of graph edges decreases from 765 to approximately 400 edges, and from 400 to 99 edges, the spectral norm starts to increase dramatically, getting very close to 1.

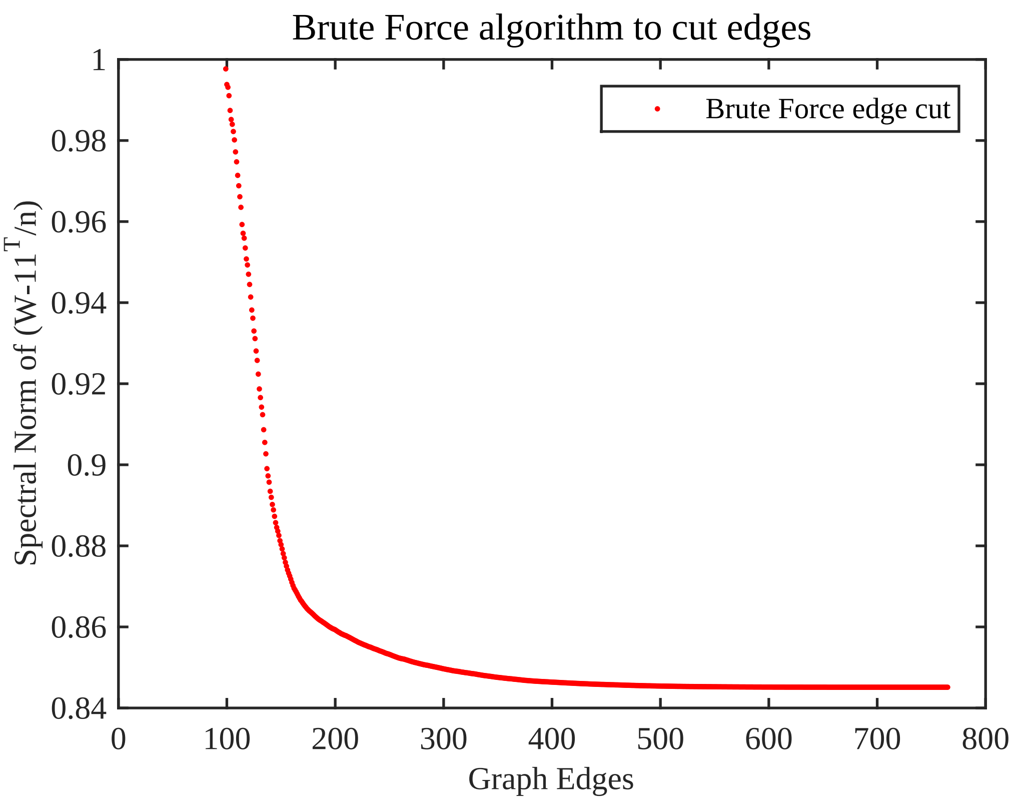


Figure 2.2: Matrix W's spectral norm given by the Brute Force algorithm.

2.3 Sparsity through random edge cuts

Formulation. Random edge cuts is a naïve approach to enforce sparsity of the communication network. However, the question “Can a network with randomly cut communication channels perform as well as a network that was optimized in structured manner?” needs to be answered.

The algorithm that answers this question is described in Algorithm 2.

If the results obtained by following this method are consistently around the greedy benchmark obtained in figure 2.2, then this method might be more efficient and surpass other more complex methods.

Algorithm 2 Random edge cuts

```
Input a network
while Current Edges $\geq$ Desired Edges do
    Remove edge;
    Check if graph remains connected;
    if True then:
        continue;
    else
        Reinstate edge;
    end if
end while
Solve FLDA for the obtained network;
```

Results. Figure 2.3 shows a comparison between the Brute Force method and the Random Edge Cut method. For the Random Edge Cut trials, 100 tests were run for each of the following number of connections: 99, 243, 266, 364, 544.

Firstly, it is possible to notice that, on the left side of the figure, the trials performed are on top of the greedy benchmark. This happens because, as the network used in this test has 100 nodes, the 99 connections correspond to a tree network. This is the worst performing network, as it uses the least possible amount of connections.

Secondly, there is a noticeable gap between the greedy benchmark and the trials performed for the remaining tests, which leads to the conclusion that a structured method exists that performs in a superior way to these Random Edge Cut tests.

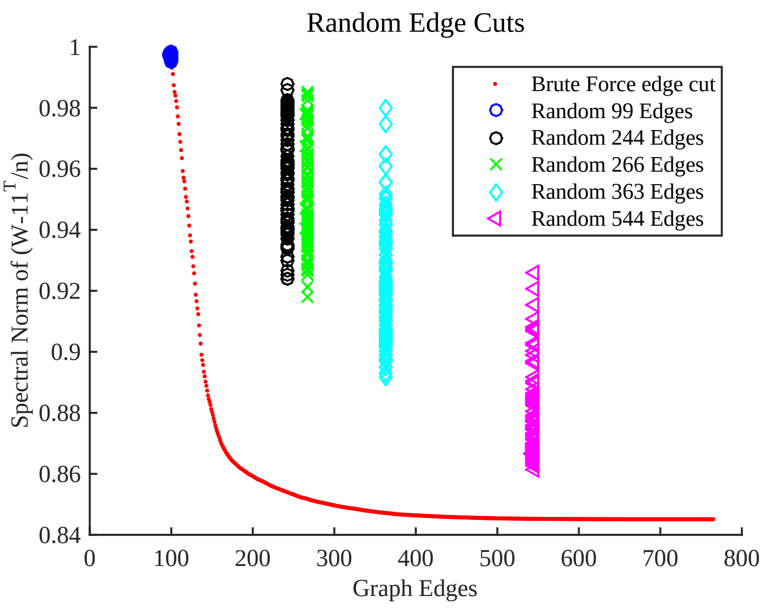


Figure 2.3: Plot of the spectral norm values obtain through 100 random cut tests at different sparsity levels, and comparison with the greedy benchmark.

2.4 Sparsity by building the graph from Star Graphs

Star graphs. A concept that exists in graph theory is that of a star graph. This type of graph is made up of one internal node, and k nodes connected to it. The central node in a star graph is critical. If this node fails, the graph becomes disconnected. Occasionally, bigger graphs can be made of compositions of several star graphs, which leads to the question "Are there nodes in the network that have a greater importance than others?".

Formulation. The initial idea was to build an entire graph as a set of star graphs. Given that the edges' weights have a meaning, the importance of the edge, it is legitimate to think that the weight of a node also carries how important it is in the network.

The first step is to optimize of the entire network, followed by the selection of the agents based on their self-weight, w_{ii}, as center of a star graph . This method serves as a way to try and understand the importance of each node in the complete graph, the same way the weight of each edge relates with its importance in the Maximum Spanning Tree method.

Results Four criteria were used to sort the agents to build the network. The first was to sort the w_{ii} values in ascending and descending order. The second was to sort these values by their magnitude, again in ascending and descending order. The results obtained are shown in Table 2.2.

Table 2.2: Cardinality of matrix W and Spectral Norm

Sorting	Cardinality	Spectral Norm	Connections
Ascending	900	0.9743	400
Descending	714	0.935	307
Magnitude Ascending	984	0.9127	442
Magnitude Descending	918	0.9218	409

It is possible to observe that sorting the nodes' self-weights in descending order achieves a lower number of connections, but it also translates into a higher spectral norm. Ordering the self-weights by the absolute value in ascending order reveals the network with most connections and best performance. A good way to evaluate the performance of these methods — to properly check if they achieve good results — is to compare them against the Random Edge Cuts method for the same number of connections as Figure 2.4 shows. It is possible to observe that three Random Edge Cuts beat the performance of the Star Graph method. This means that this method isn't better than a random choice that selects which connections should be active.

Finally, Figure 2.5 shows the networks obtained with the aforementioned method. It is definitely possible to see that the networks are less cluttered than the starting network, with less connections active. However the problem of maintaining a good performance remains, and a new approach is required.

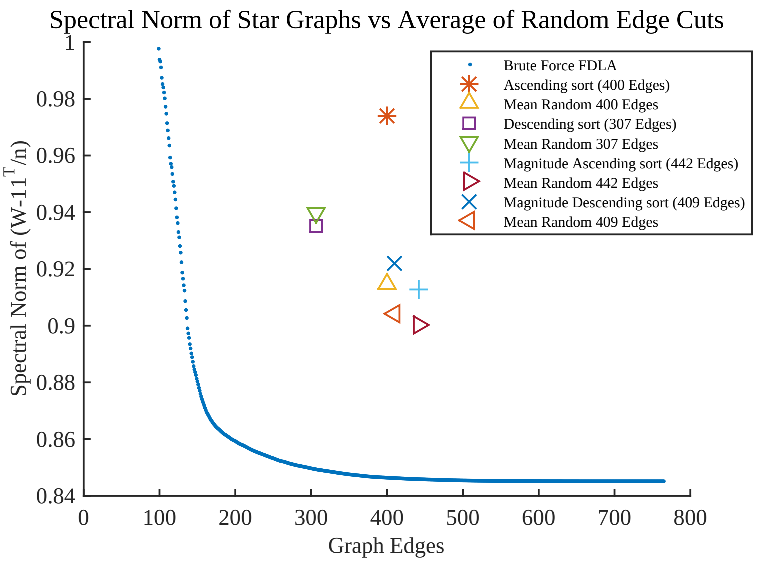
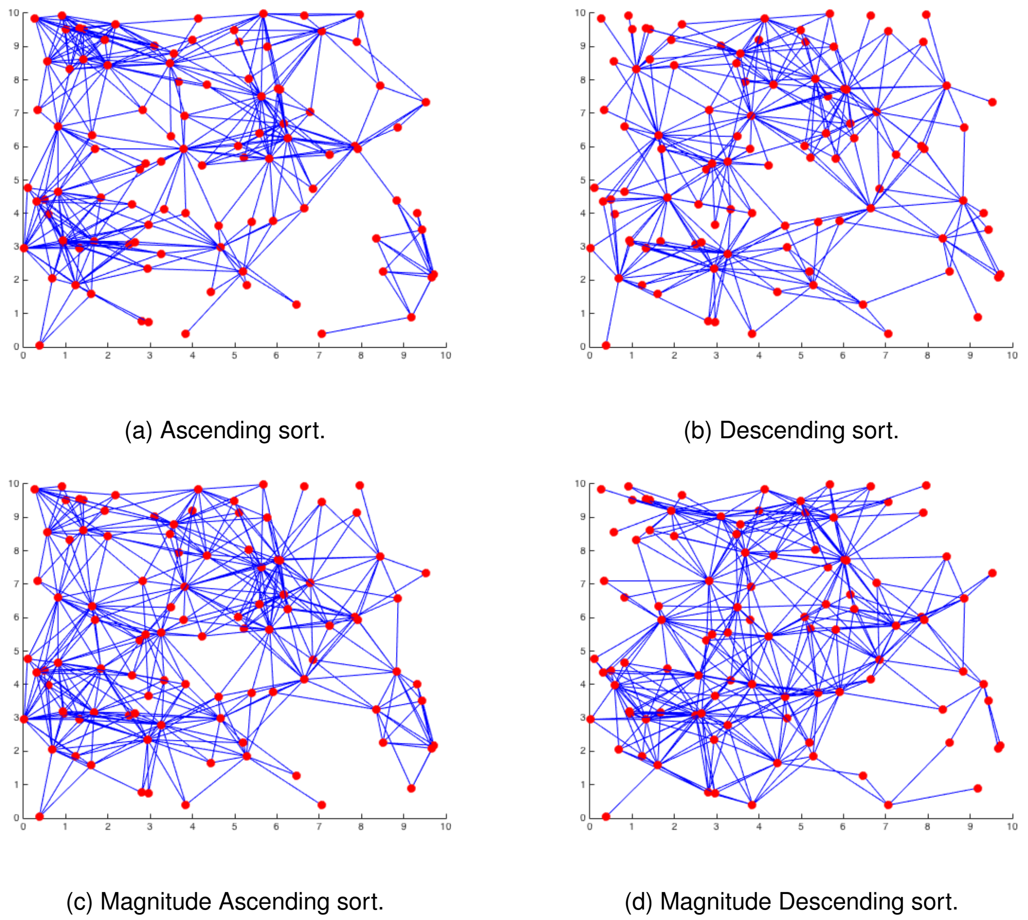


Figure 2.4: Average value of the spectral norm obtained through Random Edge Cuts compared with the spectral norm obtained with the Star Graphs method.



(a) Ascending sort.

(b) Descending sort.

(c) Magnitude Ascending sort.

(d) Magnitude Descending sort.

Figure 2.5: Observed networks formed by star graphs

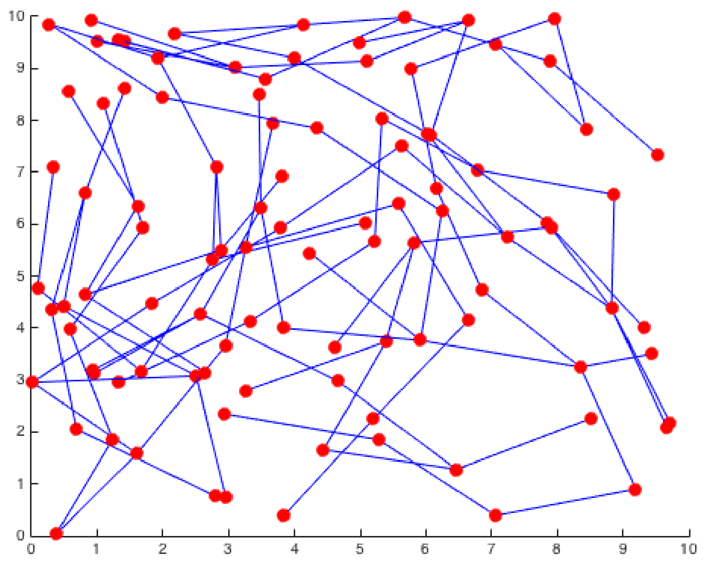
2.5 Sparsity through the Maximum Spanning Tree

Formulation. A structured technique to extremely reduce the number of communication channels active is to use the Maximum Spanning Tree of a graph after obtaining the matrix W that results from solving problem (2.18). A Tree in graph theory is a connected acyclic undirected graph, where two nodes are connected by exactly one path. The Maximum Spanning Tree is the Tree that carries the edges with the biggest weight. This work approaches the MST technique as it gives an interpretation to the weights in matrix W. Looking at matrix W, a possible interpretation is that the edges with the biggest weights are the most important edges.

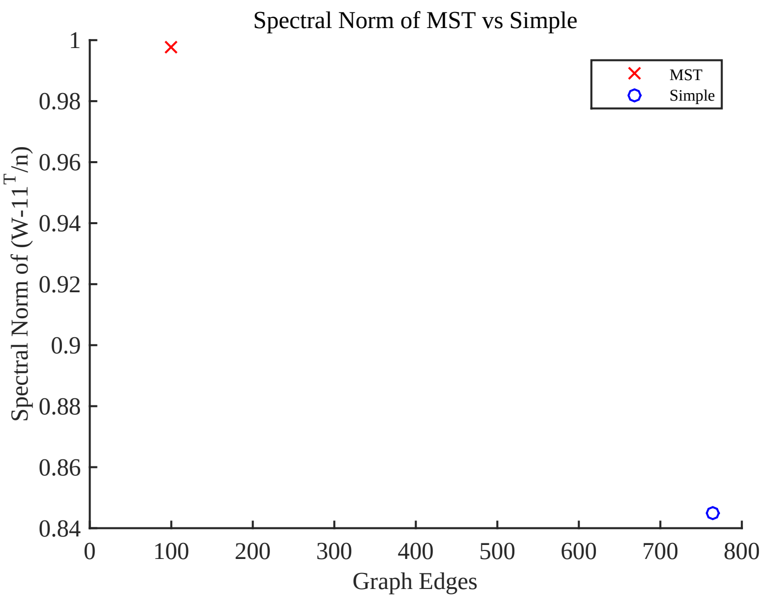
The algorithm used is simple. Firstly, the FDLA problem for the starting network is solved. Afterwards it is possible to find the Maximum Spanning Tree for the considered network. Finally, the FDLA problem for the Maximum Spanning Tree that was calculated can be solved.

Results. The MST has the minimum number of connections a graph can have. Since the graph used in this study had 100 nodes it is natural that 99 connections were obtained, as Figure 2.6a shows. The calculated matrix W, that corresponds to the MST, is the sparsest and has the biggest spectral norm obtained from a deterministic method, which can be seen in Figure 2.6b and Table 2.3.

A couple of questions remain unanswered, "Is the Maximum Spanning Tree the sparsest network with the best performance?", and "Is the Maximum Spanning Tree a good starting point for the network when edges need to be added?". The best way to answer these questions is to perform a set of random tests and compare the results to the deterministic method.



(a) Graph obtained through the MST method.



(b) Comparison of matrix W's spectral norm and number of edges used obtained from the FDLA problem and the MST algorithm

Figure 2.6: Maximum Spanning Tree results.

Table 2.3: Cardinality of matrix W and Spectral Norm of the MST weight matrix.

Cardinality	Spectral Norm	Connections
298	0.9980	99

Figure 2.7 answers the first question: the Maximum Spanning Tree does not have the best performance out of all the spanning trees that are possible. In fact, the Maximum Spanning Tree performs worse than the average obtained from the random tests.

However, things change once the spanning tree is only considered to be a starting point to add edges to. Observing Figure 2.8, it shows the evolution of the spectral norm as edges are gradually added to the graph via the application of the heuristic method described in Algorithm 3. It is possible to compare how a graph starting from the Maximum Spanning Tree evolves against graphs grown from random trees. The conclusion drawn is that the graphs grown from the Maximum Spanning Tree achieve better performances than graphs started from other trees. A possible explanation for this is that the most important channels are preserved, and as the graph grows, it only adds edges that help the network the most.

2.6 Convex L1-norm Problem

Formulation. In this version of the Fast Distributed Linear Averaging problem the l_1-norm was used to penalize non-sparse solutions. This means the objective was to minimize the number of connections each node used, *i.e.*, out of the n possible connections a node might have, it should only have $k < n$

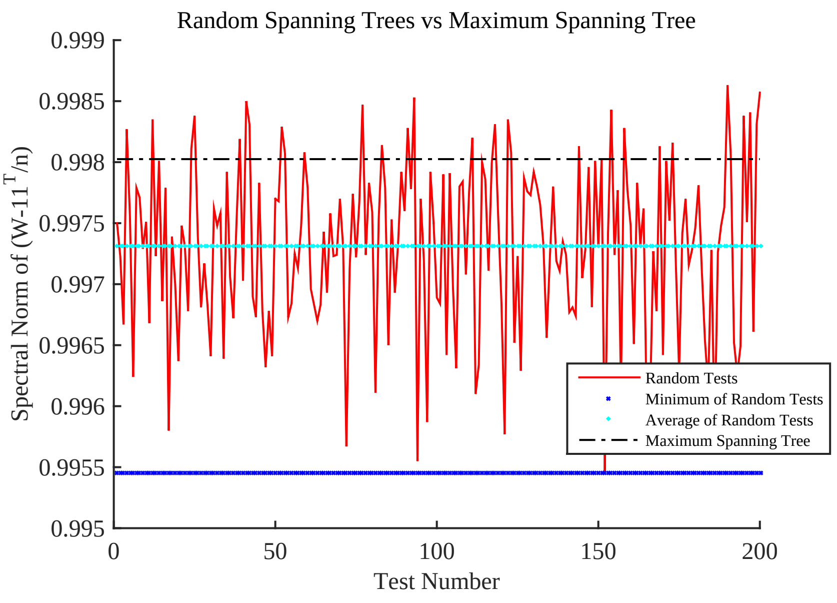


Figure 2.7: Spectral norm of random spanning trees.

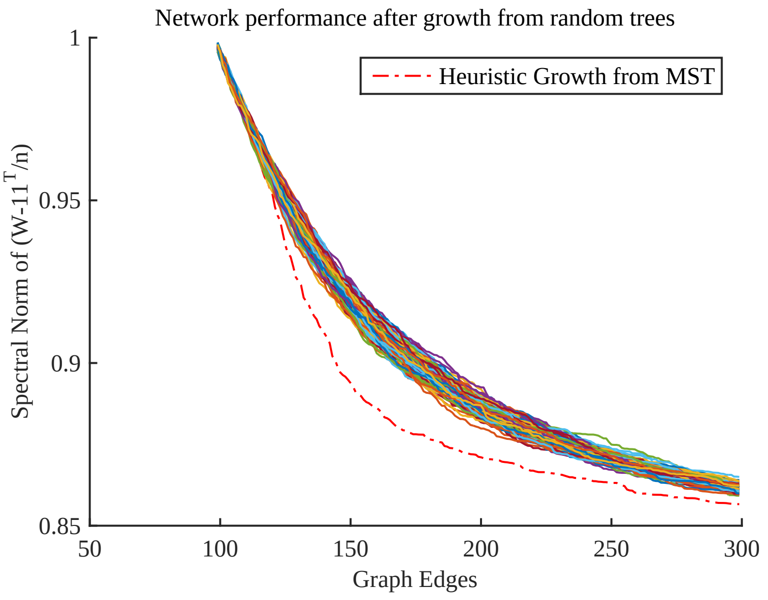


Figure 2.8: Evolution of networks grown from MST and random spanning trees.

connections active. In this context, the l_1-norm assumes the following form:

$$||W||_1 = \sum_{i,j=1}^{n} |w_{ij}|. \tag{2.19}$$

Visually, it is possible to see through Figure 2.9 that, as w_{ij} moves away from zero, it will increasingly add cost to the l_1-norm term. It is possible to argue that using w_{ij}^2 as a sparsity inducing term would be better, as it adds a bigger cost for w_{ij}'s that are far from zero, and that $|w_{ij}|$ is not a continuous function. However, when the weights w_{ij} are close to zero, the variation they produce and magnitude of the cost function w_{ij}^2 is smaller than $|w_{ij}|$. There are also techniques in Convex Optimization that allow the use

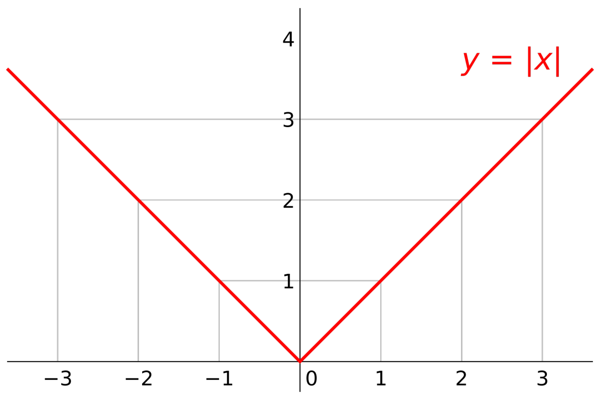


Figure 2.9: Absolute value function of w_{ij}, $|w_{ij}|$.

of non-differentiable functions.

Therefore, the optimization problem can be defined as the simple FDLA problem in equation (2.18) with the l_1-norm term added to the objective function,

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \|W - \frac{11^T}{n}\|_2 + \beta\|W\|_1 \\ \text{subject to} \quad & W = W^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{aligned} \tag{2.20}$$

Results. Firstly, notice in Figure 2.10 that, as β starts increasing, the spectral norm increases as well. This happens because a bigger β value increases the penalization associated with each non-zero entry of matrix W and, as they are brought to zero, the network's performance worsens.

Secondly, due to the constraint on W shown in equation (2.8), the l_1-norm's lower bound is the number of agents in the network. That explains the observed behaviour where, as β increases, the l_1-norm approximates 100. When β is big enough to allow this to happen, the penalizing term loses its meaning, as it becomes a constant in the objective function. The resulting W matrices will all be the same in terms of performance and all w_{ij}'s will assume positive values. As a result, this approach has limited sparsity inducing capabilities.

Table 2.4 and Figure 2.11 show the results obtained by this method. It is clearly observable that, as β increases, the network becomes less cluttered with active communication channels.

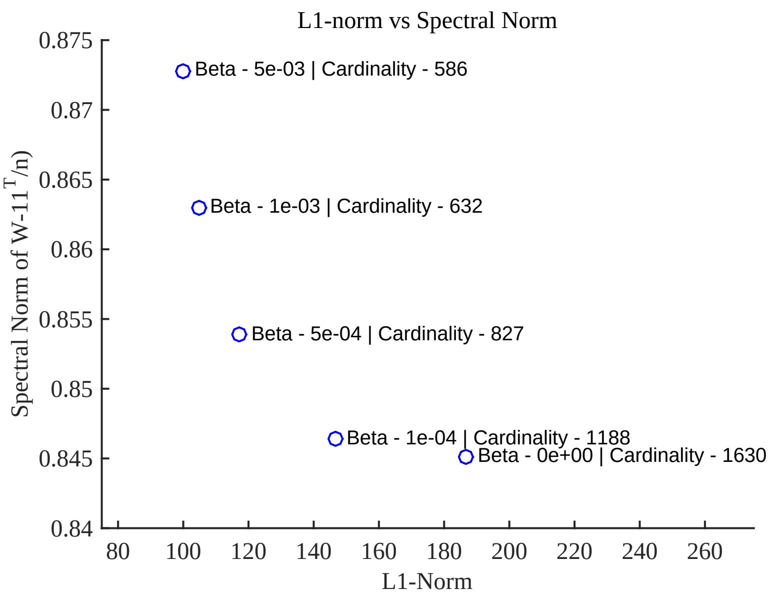
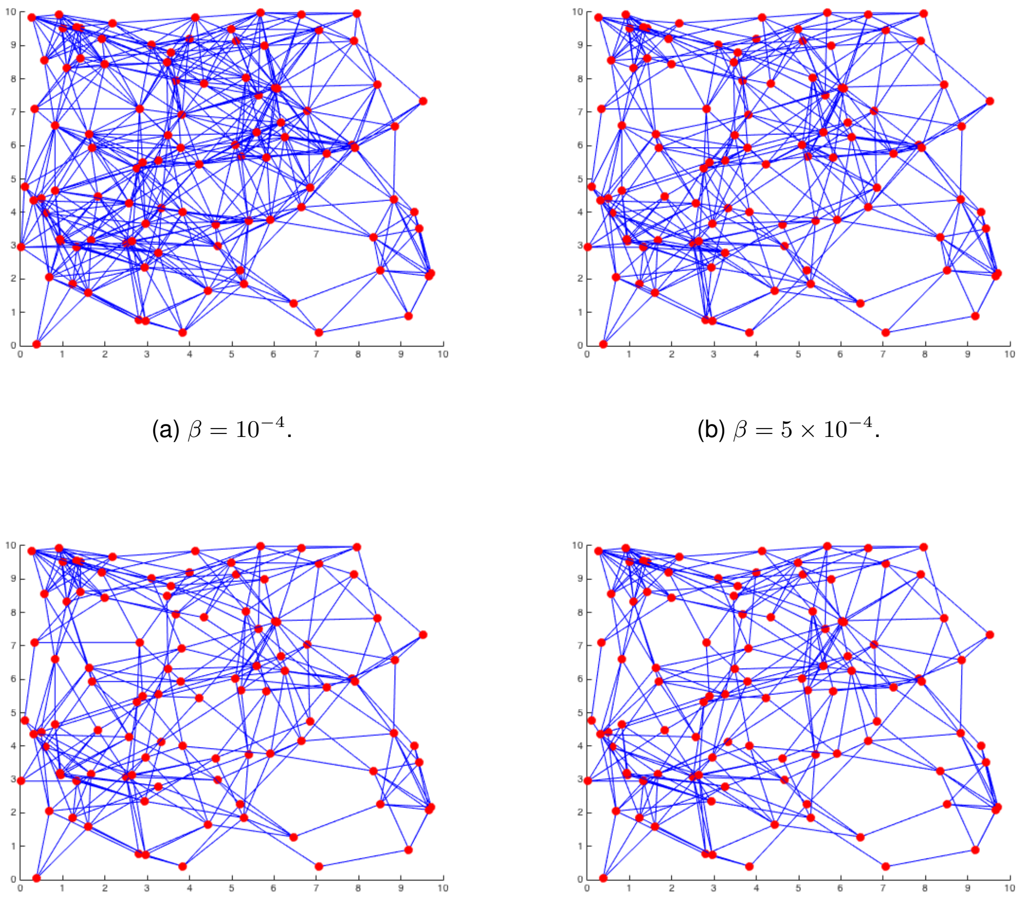


Figure 2.10: Plot of l_1-norm against matrix W's spectral norm, for increasing β values.



(a) $\beta = 10^{-4}$.

(b) $\beta = 5 \times 10^{-4}$.

(c) $\beta = 10^{-3}$.

(d) $\beta = 5 \times 10^{-3}$.

Figure 2.11: Observed sparsity increase in networks with different β.

Table 2.4: Cardinality of matrix W and Spectral Norm

Beta	Cardinality	Spectral Norm	Connections
10^{-4}	1188	0.8464	544
5×10^{-4}	827	0.8538	363
10^{-3}	632	0.8630	266
5×10^{-3}	586	0.8728	243

2.7 Convex re-weighted l_1-norm Problem

Formulation. The re-weighted l_1-norm is more powerful than the l_1-norm as it will penalize with a higher cost entries that are far from zero, thus making it an important tool to further minimize the communication between nodes.

The re-weighted l_1-norm formulation uses the logarithm due to the fact that, as its argument goes to zero, it strongly tends to $-\infty$. This forces entries to stay close to zero as a slight variation will change the cost a lot more than simply using absolute value. The re-weighted l_1-norm has the following form:

$$\log(|w_{ij}| + \varepsilon), \tag{2.21}$$

where ε is a slight translation ($\approx 10^{-6}$) so that the function is defined when $w_{ij} = 0$. It also provides $w_{ij} = 0$ with a fixed cost equal to $\log(\varepsilon)$. However, if (2.21) was added to formulation (2.18) it would no longer be a problem in the Convex Optimization domain, as the formulation would become non-convex. Therefore a second step is needed to make this equation suitable to enter the Convex Optimization process — linearizing equation (2.21) by using the first order Taylor expansion, which results in:

$$\frac{|w_{ij}|}{|w_{ij}^{(0)}| + \varepsilon} + \text{cte} \tag{2.22}$$

It is now possible to define an algorithm to calculate matrix W as, by analysis with convex algebra, it is possible to conclude that equation (2.22) is convex. Each new entry of $W^{(t+1)}$ will be an entry of the matrix W that minimizes the problem:

$$\begin{aligned} W^{(t+1)} := \underset{W}{\text{argmin}} \quad & \left\| W - \frac{11^T}{n} \right\|_2 + \sum_{i,j} \frac{\beta}{|w_{ij}^{(t)}| + \varepsilon} |w_{ij}| \\ \text{subject to} \quad & W = W^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{aligned} \tag{2.23}$$

Optimization process (2.23) differs from the previous formulations as it is an iterative algorithm and, as such, needs to be seeded with an initial matrix $W^{(0)}$, obtained from the solution of problem (2.18). Being an iterative algorithm it needs a stopping criterion, which was defined as:

$$\frac{\left\| W^{(t+1)} - W^{(t)} \right\|}{\left\| W^{(t)} \right\|} \leq 0.05. \tag{2.24}$$

This criterion means that the algorithm stops as soon as the variation in the matrices between two consecutive iterations is smaller than 5%.

Results. The first results obtained pointed to a problem in the formulation given by equation (2.23). In that form, the re-weighted l_1-norm was too strong as a sparsity inducing term, and the resulting communication networks were disconnected and had a spectral norm of 1. When the spectral norm is equal to 1, the resulting matrix W is not suitable to solve the consensus problem, because it will not converge to the average.

A new constraint was added to equation (2.23), forcing the spectral norm to be within a given margin,*e.g.* 1%, giving the new formulation:

$$\begin{aligned} W^{(t+1)} := \underset{W}{\text{argmin}} \quad & \left\| W - \frac{11^T}{n} \right\|_2 + \sum_{i,j} \frac{\beta}{|w_{ij}^{(t)}| + \varepsilon} |w_{ij}| \\ \text{subject to} \quad & W = W^T \\ & W1 = 1 \\ & \left\| W - \frac{11^T}{n} \right\|_2 \leq \left\| W^{(0)} - \frac{11^T}{n} \right\|_2 \times (1 + 1\%) \\ & W \in \mathcal{S}. \end{aligned} \tag{2.25}$$

This slight variation provides results capable of solving the consensus problem. Two formulations of problem (2.25) were considered: one where the diagonal entries of the matrix have a cost ($i = j$ is allowed), and one where they do not ($i \neq j$ is imposed).

Because it is an iterative problem, this formulation takes a lot longer to calculate than the previous formulations, however it also yields better results, as can be observed in Table 2.5.

Table 2.5: Cardinality of matrix W and Spectral Norm with allowed diagonal entries

Allowed margin	Spectral Norm	Connections
1 %	0.8519	279
2 %	0.8586	225
5 %	0.8779	165

Figure 2.12 shows the results obtained with several tolerances in the spectral norm constraint. The data labeled *diag* $\alpha\%$ corresponds to the formulation that considers the diagonal entries, and the data labeled *nodiag* $\alpha\%$ relates to the formulation that does not consider diagonal entries.

An important remark to make is that the final decrease in spectral norm comes from re-optimizing the final network with the FDLA formulation. Since the FDLA formulation is cost-blind, it will use every possible connection and self-weight optimally. This decrease happens in the *diag* $\alpha\%$ formulation because the re-weighted l_1-norm also takes into account the diagonal entries of matrix W, which are the self-weights of each node, and tries to set them to zero, thus worsening the performance. In the *nodiag* $\alpha\%$ the diagonal entries have no cost, but still the simple FDLA formulation can achieve a small increase in performance.

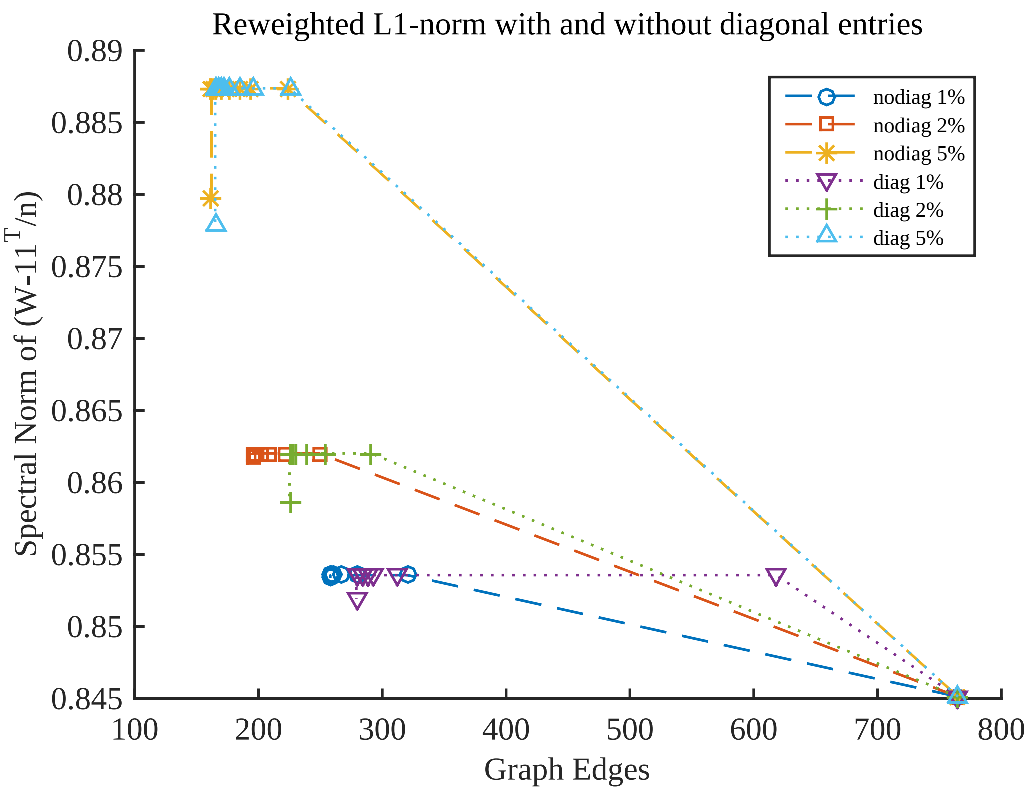


Figure 2.12: Spectral norm and number of communication channels active after the re-weighted l_1-norm problem.

2.8 Growing Well-connected Graphs

This part of the work explores how to grow graphs. The idea behind growing a graph is that, starting from a simple network, one wants to add the edges that create a network that achieves the best performance in solving the consensus problem. Three methods were explored for graph growth: maximizing the Algebraic Connectivity, minimizing the graph's Average Path Length, and using the dual variables of the solution of the FDLA problem.

Algebraic Connectivity. The second smallest eigenvalue of the Laplacian matrix is called the algebraic connectivity of the graph. It is used as a measure of how well-connected a graph is [13], where a bigger value means a better connectivity, *i.e.* if graph $\mathcal{G}_\mathsf{A}$ is contained in graph $\mathcal{G}_\mathsf{B}$, $\mathcal{G}_\mathsf{A} \subseteq \mathcal{G}_\mathsf{B}$, then $\lambda_2(L_{\mathcal{G}_\mathsf{A}}) \leq \lambda_2(L_{\mathcal{G}_\mathsf{B}})$.

1st Problem statement. Boyd formulated a problem where the objective was to grow a graph to a desired number of connections [13]. This was done under the premisse of maximizing the final connectivity, by choosing the best possible edges to do so.

The problem starts with a base Laplacian, $\mathbf{L}_\text{base}$ and a set of m edges from which $k \leq m$ edges are choosen and added to the graph. The question to answer is "What is the best set of edges to choose and add?".

To respond to this question, it is important to start by defining the incidence matrix A. This matrix contains, as column vectors, all the edges present in an undirected graph. Edge l connecting nodes i and j is defined as a vector $a_l \in \mathbf{R}^n$ with entry $a_{l_i} = 1$, $a_{l_j} = -1$ and all other entries as 0. The collection of all these vector creates the incidence matrix $A \in \mathbf{R}^{n\times m}$. The Laplacian can then be expressed as:

$$\mathbf{L}_{\text{base}} = AA^T = \sum_{l=1}^{m} a_l a_l^T. \tag{2.26}$$

This characteristic allows formulation (2.27) where, given a base Laplacian, a search is made to find the edges that maximize λ_2.

$$\begin{aligned} \max_{\lambda_2} \quad & \lambda_2(\mathbf{L}_{\text{base}} + \sum_{l=1}^{k} a_l x a_l^T). \\ \text{subject to} \quad & 1^T x = k \\ & x_i \in \{0,1\}, \quad \text{for } i = 1,\dots,l. \end{aligned} \tag{2.27}$$

This is a boolean problem where x is the vector that selects the edges to add. Though it is solvable with an exhaustive search, it is not scalable for big graphs. The amount of times that $\lambda_2(\mathbf{L})$ would need to be calculated is $\binom{m}{k}$.

This work is not interested in exhaustive searches and, as such, problem (2.27) can be relaxed into one where x_i ranges between 0 and 1. This relaxation leads to problem (2.28) which can now be solved by Convex Optimization.

$$\begin{aligned} \max_{\lambda_2} \quad & \lambda_2(\mathbf{L}_{\text{base}} + \sum_{l=1}^{k} a_l x a_l^T). \\ \text{subject to} \quad & 1^T x = k \\ & 0 \le x_i \le 1, \quad \text{for } i = 1,\dots,l. \end{aligned} \tag{2.28}$$

If problem (2.28) is solved with a binary x vector, then that solution is also optimal for problem (2.27). If not, the largest k entries of vector x can be set to 1 and the remaining to 0, thus being a sub-optimal solution.

This problem can be formulated as an Semidefinite Programing problem which can be easily solved. The first step is to introduce the Reileigh-Ritz method to find λ_2: $\lambda_2 = \inf_x\{q^T\mathbf{L}(x)q \mid \|q\| = 1, \mathbf{1}^T q = 0\}$, with $\mathbf{L}(x) = \mathbf{L}_{\text{base}} + \sum_{l=1}^{k} a_l x a_l^T$.

Observe that $L(x)\mathbf{1}/\sqrt{n} = 0$. As such, when the eigenvalue decomposition of $L(x)$ is made, the resulting matrices are:

$$\begin{aligned} L(x) &= \begin{bmatrix} U & \mathbf{1}/\sqrt{n} \end{bmatrix} \begin{bmatrix} \lambda_n & & & \\ & \ddots & & \\ & & \lambda_2 & \\ & & & 0 \end{bmatrix} \begin{bmatrix} U^T \\ \mathbf{1}^T/\sqrt{n} \end{bmatrix} \\ &= U \begin{bmatrix} \lambda_n & & \\ & \ddots & \\ & & \lambda_2 \end{bmatrix} U^T + \mathbf{1}/\sqrt{n} \times 0 \times \mathbf{1}^T/\sqrt{n} \end{aligned} \tag{2.29}$$

It is a well known fact in convex optimization that, when the objective is to maximize the smallest eigenvalue, the convex form of $\lambda_{\text{min}}(A) \geq t$ is $A \succeq t\, I_n$. So, with the eigenvalue decomposition of $L(x)$, the aim is to search for the convex form of maximizing the eigenvalue $\lambda_2(L(x))$.

Firstly, to say that $\lambda_2(L(x)) \geq s$, is the same as saying $\lambda_n \geq \ldots \geq \lambda_2 \geq s$ and it is possible to express this by a matrix inequality,

$$\begin{aligned} \begin{bmatrix} \lambda_n & & \\ & \ddots & \\ & & \lambda_2 \end{bmatrix} \succeq \begin{bmatrix} s & & \\ & \ddots & \\ & & s \end{bmatrix} &\Leftrightarrow U \begin{bmatrix} \lambda_n & & \\ & \ddots & \\ & & \lambda_2 \end{bmatrix} U^T \succeq U \begin{bmatrix} s & & \\ & \ddots & \\ & & s \end{bmatrix} U^T \\ &\Leftrightarrow L(x) \succeq s\, UU^T. \end{aligned} \tag{2.30}$$

Equation $L(x) \succeq t\, UU^T$ is very close the convex optimization form $A \succeq t\, I_n$. It is important do figure out how to describe UU^T in terms of known matrices. From the eigenvalue decomposition:

$$\begin{aligned} \begin{bmatrix} U & \mathbf{1}/\sqrt{n} \end{bmatrix} \begin{bmatrix} U^T \\ \mathbf{1}^T/\sqrt{n} \end{bmatrix} = I_n &\Leftrightarrow UU^T + \frac{\mathbf{1}}{\sqrt{n}} \frac{\mathbf{1}^T}{\sqrt{n}} = I_n \\ &\Leftrightarrow UU^T = I_n - \frac{\mathbf{1}\mathbf{1}^T}{n} \end{aligned} \tag{2.31}$$

Finally, joining (2.30) and (2.31) the SDP formulation of problem (2.28) appears:

$$\begin{aligned} \max_{s} \quad & s \\ \text{subject to} \quad & s\left(I - \mathbf{1}\mathbf{1}^T/n\right) \preceq L(x) \\ & \mathbf{1}^T x = k \\ & 0 \leq x_i \leq 1, \quad \text{for } i = 1, \ldots, l. \end{aligned} \tag{2.32}$$

Though SDP's can easily be solved, the computational requirements of this approach motivated Boyd to also develop an heuristic approach to this problem. The greedy heuristic developed is an iterative algorithm, instead of a one shot method like the SDP.

The heuristic algorithm is based on the eigenvector associated with the second smallest eigenvalue, the Fiedler vector, and is structured as shown in Algorithm 3.This is a greedy algorithm because it looks at all edges each iteration and chooses the best one. Thus, it only guarantees the best result possible at each iteration and not the best overall result.

A modified version of this algorithm was also implemented based on the rollout idea [14], trying to improve its results. The rollout method starts by adding an edge and then uses the greedy algorithm to predict what edges to add. Then, it scores the first edge that it added based on the prediction score. This prediction acts as a capacity to “look ahead” and chose the best path for the long term. Although it is computationally more intense, it is usually a method that returns much better results.

Average Path Length. Graph theory introduces a simple concept called diameter of a graph. It describes the longest path between any two nodes of the graph. The diameter of a graph can be interpreted as the number of iterations needed for the information of one node to reach the node that is farthest.

The diameter of a graph is the maximum entry of the matrix that stores the distances between all nodes. This path matrix can be constructed with the application of the Floyd-Warshall algorithm.

Algorithm 3 Greedy Heuristic algorithm to add k edges.

```
Input a Laplacian;
while desired number of edges not reached do
    Calculate Fiedler vector, $v$, of the Laplacian;
    for each edge in the set do
        Determine: $score = (v_i - v_j)^2$;
    end for
    Add edge with the highest $score$;
    Update Laplacian;
end while
```

2nd Problem Statement. The problem now at hand focuses on choosing the edge that minimizes the distance that the information must travel. For this to happen, three methods were designed.

The first method was simply to choose the edge that provided the largest decrease in graph diameter. This choice was indiscriminate, so that it would be possible to evaluate how important the diameter of the network was. This heuristic is a coarse estimate on how well a network might perform when it comes to solving the consensus problem.

The second method was an improvement on the first one. The idea was not to choose a random edge, from all of those that delivered the largest decrease in graph diameter, but to weight each choice against each other by seeing which edge also decreased the Average Path Length the most.

Finally, the third method was to simply chose the edge that delivered the largest decrease in the Average Path Length.

All these methods are applied via an iterative algorithm, where the Floyd-Warshall algorithm is applied at each iteration see how each edge changes the distance matrix.

Dual Problem and Dual Variables. As explained in Boyd's "Convex Optimization" [11], each problem solved using Convex Optimization has associated with it a dual problem. Dual problems appear when a simplification is made through the Lagrangian of the primal problem. When the Lagrangian is calculated, dual variables are introduced, and they relate with each constraint considered in the Lagrangian. The basic idea in Lagrangian duality is to take the constraints in (1.9) into account by augmenting the objective function with a weighted sum of the constraint functions.

Consider the standard form of an optimization problem, (1.9). As stated, the Lagrangian is calculated by taking into account the constraints on x and the cost function $f_0(x)$, and assumes the following form:

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x). \tag{2.33}$$

Variable λ_i is the Lagrange multiplier associated with the i^{th} inequality constraint $f_i(x) \leq 0$; and similarly ν_i is the Lagrange multiplier associated with the i^{th} equality constraint $h_i(x) = 0$.

The Lagrange dual function is defined as the minimum value of the Lagrangian over x, for $\lambda \in \mathbf{R}^m$

and $\nu \in \mathbf{R}^p$, and is given by:

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = \inf_x \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right), \tag{2.34}$$

where function $g(\lambda, \nu)\ :\ \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}$. The dual function is the pointwise infimum of a family of affine functions of λ and ν, it is concave, even when the problem is not convex. Therefore the dual problem is the maximization of $g(\lambda, \nu)$, expressed as:

$$\begin{aligned} &\max_{\lambda,\nu} && g(\lambda, \nu) \\ &\text{subject to} && \lambda_i \geq 0, \quad \text{for } i = 1, \cdots, m. \end{aligned} \tag{2.35}$$

3rd Problem Statement. A possible interpretation for the Dual Variables is how sensitive the cost function is to a variation in a respective constraint. Knowing this, it is possible to try and reformulate the brute force algorithm to a more intelligent and less computationally expensive algorithm to grow graphs.

Consider the graph obtained via MST method. The connections that are not active belong in two categories, the impossible connections and the connections that are turned off. After the optimization of the spectral norm associated with matrix W, each of those constraints will have a dual variable that relates how sensitive the spectral norm would be to a change in the status of the connection, from off to on. The idea is to turn on, at each iteration, only the communication channel that will have the most impact, and that is not one of those that represent an impossible channel. The aforementioned idea is described in algorithm 4.

Algorithm 4 Algorithm with Dual Variables to add edges.

```
Input a Laplacian;
Perform optimization on the $\rho(W - \mathbf{1}\mathbf{1}^\mathbf{T}/n)$
while desired number of edges not reached do
    Find the $\min$ of the dual variables associated with possible communication channels;
    Turn that communication channel “on”;
    Update Laplacian;
    Perform optimization on the $\rho(W - \mathbf{1}\mathbf{1}^\mathbf{T}/n)$;
end while
```

Results. Since this part of the work focuses on growing graphs, a starting network needs to be chosen. From the previous formulations, it is possible to conclude that a good starting point is the Maximum Spanning Tree, as it shows the minimum number of active connections, and can be obtained after solving two simple FDLA problems, (2.18).

Once again, the first naive approach to growing a graph is to do a brute force method. The algorithm is similar to the one shown before, but instead of removing an edge and testing the network, an edge is added from the set of possible edges. This method also represents the best possible growth, since at each iteration all edges are tested, and only the one that provides the best performance is added. Evidence of this is the small gap between the curves in figure 2.13.

Figure 2.14 pictures the behaviour of the SDP formulation based on the connectivity of the graph. It is possible to see how the one shot version (k = #edges to add) compares with the iterative version ($k = 1$). The iterative version approximates the optimal curve, while the one shot method fails to do so. The strange behaviour that is observed in the one shot method appears when the variable x, that is used as a vector to select which edges to add, is converted from a vector with continuous entries to one with discrete values. As stated, the relaxation that is shown in problem (2.32) can only produce sub-optimal solutions for the connectivity, which will reflect on the performance of the network. The heuristic shown in figure 2.15 is much faster than solving the SDP, and shows surprisingly good results, as the gap between its curve and the greedy benchmark is also very small.

The data in figure 2.16 shows the fortified implementation of Boyd's heuristic algorithm. The fortified implementation usually shows better results, as it scores the edge to add based on a prediction of a final score. Computationally, it is a method that requires more time than the simple heuristic method. It was observed that the fortified method did not behave significantly better than the simple heuristic, therefore being worse by the computational resources used.

Figures 2.17, 2.18 and 2.19, all show a similar behaviour to the greedy benchmark curve, although they are not as good as the results obtained by Boyd's heuristic method. The images show an evident correlation between the longest paths existent in the network and the performance obtained, but also indicate that the average path length criterium is not the most important to achieve a good performance.

Lastly, the iterative method involving the Dual is very close to the Brute Force curve, while being much less computationally expensive. The interpretation given to the dual variables proves itself as correct, as choosing which edge to add based on it's dual variable value, gave a very good result.

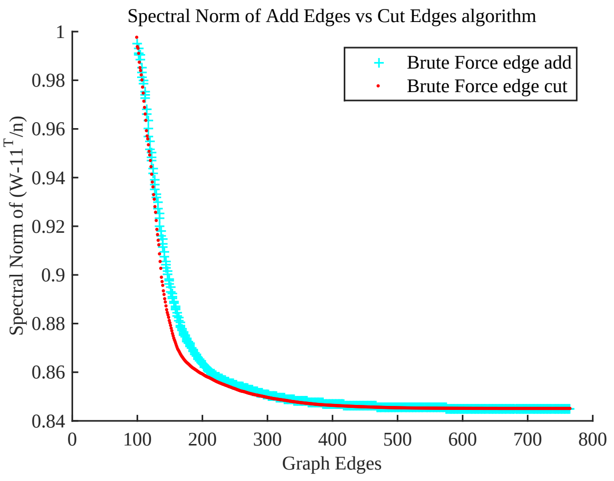


Figure 2.13: Greedy benchmark of the spectral norm compared with Brute Force growth.

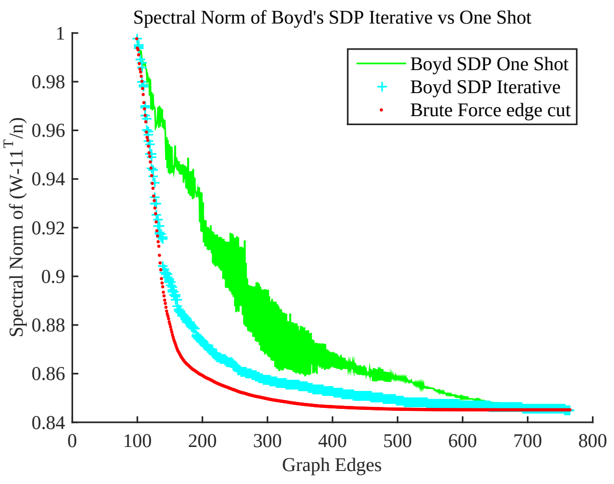


Figure 2.14: Greedy benchmark of the spectral norm compared with SDP growth

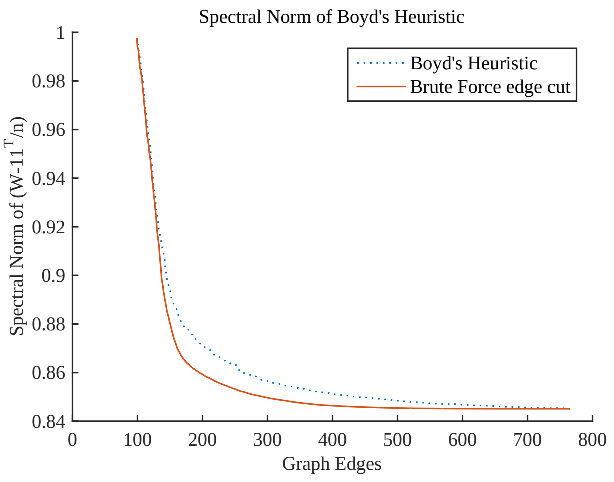


Figure 2.15: Greedy benchmark of the spectral norm compared with Heuristic growth.

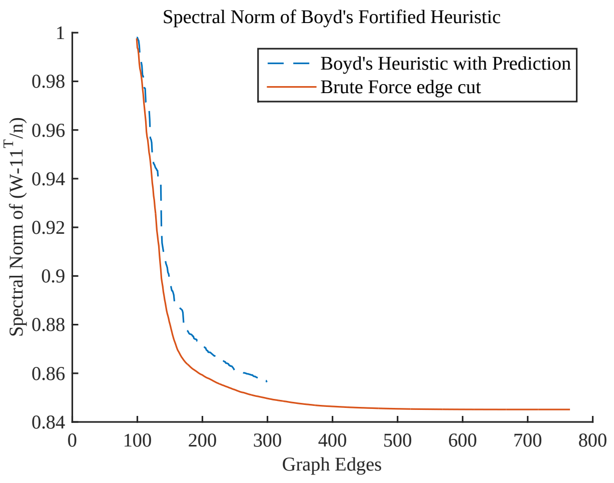


Figure 2.16: Greedy benchmark of the spectral norm compared with Fortified Heuristic growth.

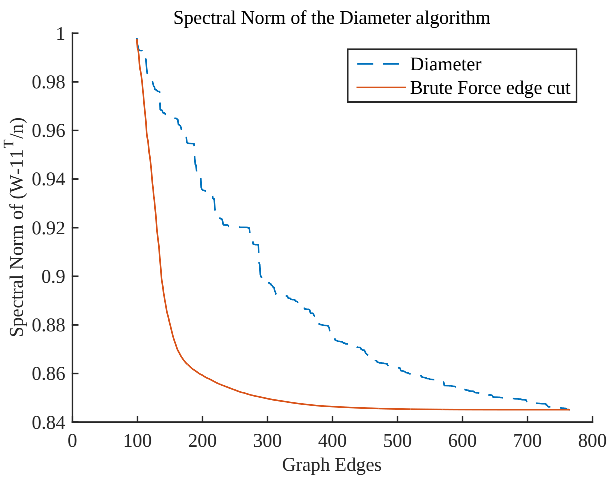


Figure 2.17: Greedy benchmark of the spectral norm compared with Diameter growth.

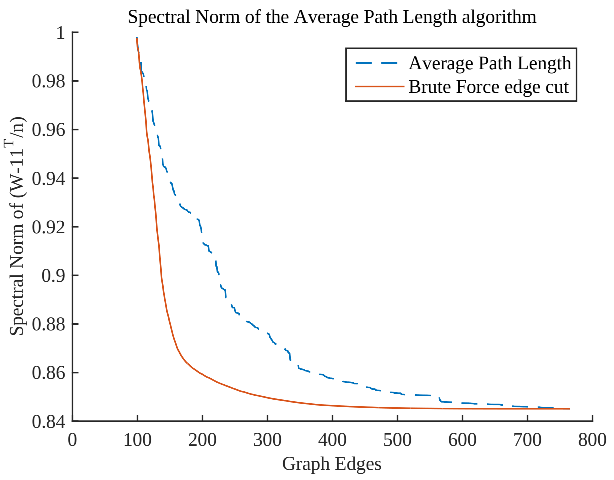


Figure 2.18: Greedy benchmark of the spectral norm compared with Average Path Length growth.

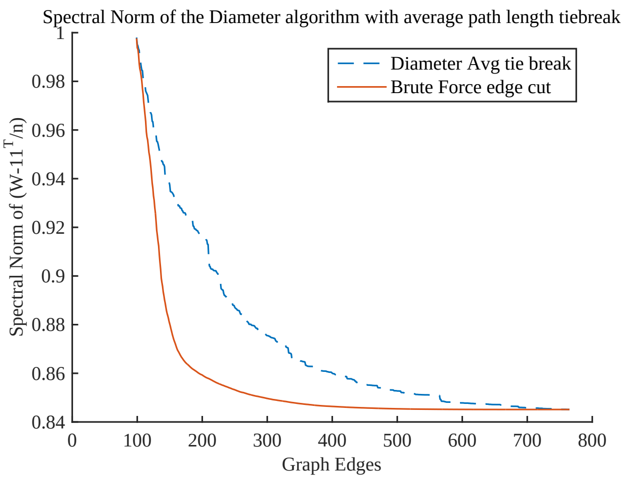


Figure 2.19: Greedy benchmark of the spectral norm compared with Diameter (2nd method) growth.

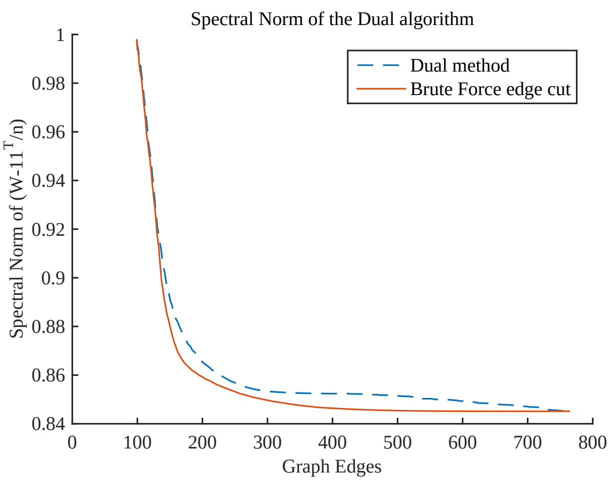


Figure 2.20: Greedy benchmark of the spectral norm compared with Dual growth.

Figure 2.21: All growth methods compared to the Greedy benchmark obtained by the Brute Force method to remove edges.

2.9 Closing Remarks

This chapter achieved some interesting results. Several of the tested methods observed sparse networks that held a good performance level, and therefore a W matrix with a small spectral norm. It is only natural to evaluate the performance of these methods against the brute force approaches that were tried, since they were the ones that tested each edge before adding or removing the best one.

First and foremost, it is noticeable that the Dual, and Boyd's SDP and heuristic follow the brute force algorithm for growth closely. This is important, as the amount of calculations needed is much lower in these algorithms and the results they achieve are good.

Secondly, these also represent the only methods that allow for some control of the number of connections that is desired, because they are iterative. The l_1-norm and the re-weighted l_1-norm problems allow for no such direct control. The first permits some fine tuning of the connections used with a well chosen β value, the variable that attributes cost to the connections. This method however can not achieve a very sparse network, as a limit is reached when the l_1-norm of W is equal to the number of nodes. The re-weighted l_1-norm method allows for no such control, as each iteration attributes cost to a connection based on the previously calculated W matrix. It achieves results as good as the iterative algorithms, as shown in Figure 2.22.

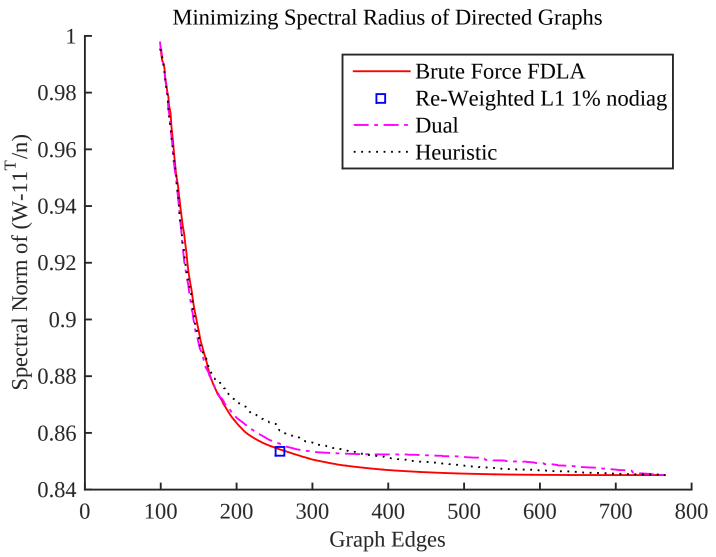


Figure 2.22: Brute Force, Dual, Heuristic and Re-weighted l_1-norm 1% no diagonal compared.

Finally, it is also worth mentioning that the re-weighted l_1-norm allows the user to control the performance penalty. The bigger the penalty allowed, the sparser the network can be.

Chapter 3

Matrix W for Directed Graphs

3.1 Introduction

Until this point, this work focused on undirected graphs. The matrices that represent them are symmetric and therefore easier to work with.

A step towards a more generic work involves moving from undirected graphs to directed graphs. In directed graphs, the channels between nodes are not bidirectional. For example, nodes i and j might only have a channel that goes from $i \rightarrow j$, making communication unidirectional. This translate into having a weight $w_{ij} \neq 0$ and $w_{ji} = 0$, which breaks the previous symmetry of matrix W.

When W is not symmetric, its (non-convex) spectral radius does not coincide with its (convex) spectral norm. Therefore, other methods are needed to design the matrix W that yields the fastest consensus.

Before moving on to the algorithms that were created, it is important to note that not all directed graphs represent networks able to reach a consensus. In order to reach, a consensus the network must be strongly connected. A strongly connected network is a network where it is possible to travel from any node i to any node j. If the network has this property, it means that information contained in any node can reach any other node in a finite number of iterations.

3.2 Minimizing the spectral norm

The spectral norm as a proxy for the spectral radius. Since the matrix W that represents a directed graph is not symmetric, minimizing the spectral norm is not the same as minimizing the spectral radius. Recall that the spectral norm of a matrix corresponds to its the maximum singular value. However, for a generic square matrix A, inequality (3.1) holds.

$$\rho(A) \leq \|A\|_2 \tag{3.1}$$

That is, the spectral radius $\rho(A)$ of an arbitrary matrix A is upper-bounded by its spectral norm $\|A\|_2$. This inequality suggests the use of the spectral norm as a surrogate for the spectral radius.

Indeed, since $\|A\|_2$ is a convex function of the entries of A, the minimization of $\|A\|_2$ over a convex set is an easy task, from the numerical viewpoint. In sum, the following convex optimization problem exists:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \left\| W - \frac{1}{n} 11^T \right\|_2 \\ \text{subject to} \quad & 1^T W = 1^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{aligned} \tag{3.2}$$

Numerical results. The formulation used is able to provide a weight matrix that reaches a consensus for a strongly connected directed graph. Since this is a simple and straightforward problem to solve, it will establish the baseline with which future results will be compared.

Table 3.1: Cardinality of matrix W and spectral radius of the weight matrix.

spectral radius	Connections
0.9263	737

To assess the quality of the approach in (3.2), its performance was compared the performance of a random search. The random search consisted in a batch of Monte Carlo experiments; each Monte Carlo projects a random matrix with independent and identically distributed standard gaussian entries—in matlab notation, $\bar{W} = \mathtt{randn(n, n)}$—onto the feasible set, i.e., it solves the convex optimization problem:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & ||W - \bar{W}||_2 \\ \text{subject to} \quad & 1^T W = 1^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{aligned} \tag{3.3}$$

These trials were preformed $2 \cdot 10^4$ times, and, for each matrix, its spectral radius was recorded. The results are displayed in Figure 3.1. There are two data points that are very important: trials 1455 and 3622 yielded a spectral radius of 0.9226 and 0.9044 respectively. These values are lower than the spectral radius of the solution of (3.2). This confirms that the approach in (3.2) does not lead to the fastest weigth matrix.

3.3 The P-Q iterations

The algorithm developed in this section was based on a series of reformulations of the problem (2.9):

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \rho(W - \mathrm{J}) \\ \text{subject to} \quad & W \in \mathcal{W}, \end{aligned} \tag{3.4}$$

where $W \in \mathcal{W}$ represents the set of constraints on the matrix W, (2.1), (2.7) and (2.8), and $\mathrm{J} = \mathbf{1}\mathbf{1}^T/n$. The reformulations result in an optimization problem that is amenable to convex optimization methods.

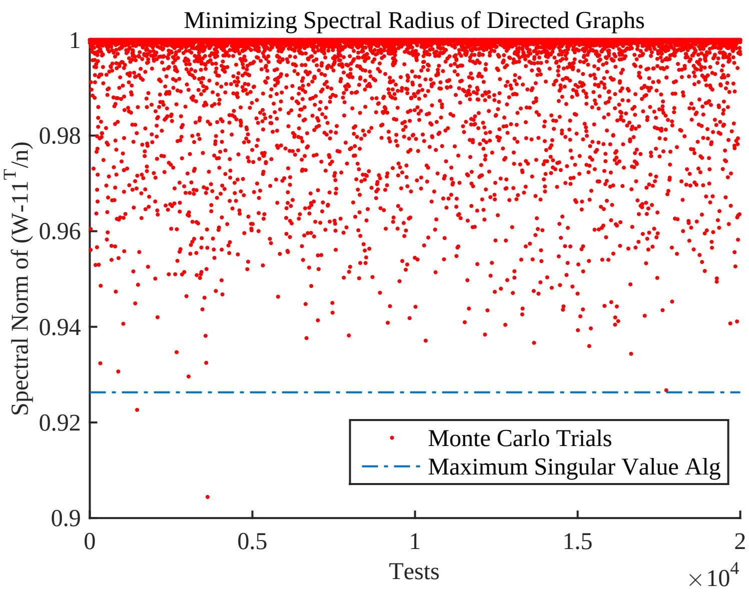


Figure 3.1: Monte Carlo trials to find W matrices that reach consensus.

Firstly, an epigraph variable is introduced and the problem switches to

$$\begin{aligned} \underset{W,t}{\text{minimize}} \quad & t \\ \text{subject to} \quad & W \in \mathcal{W} \\ & \rho(W - \mathrm{J}) \leq t \\ & t \geq 0. \end{aligned} \tag{3.5}$$

Now, the well-known fact that

$$\rho(W - \mathrm{J}) \leq t \quad \Leftrightarrow \quad \exists_{P \succ 0} : (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq t^2 P. \tag{3.6}$$

is exploited. In words, condition (3.6) says that an upper-bound on the spectral radius can be certified by a matrix inequality, involving an extra variable P.

Applying (3.6) in (3.5) gives

$$\begin{aligned} \underset{W,t,P \succ 0}{\text{minimize}} \quad & t \\ \text{subject to} \quad & W \in \mathcal{W} \\ & (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq t^2 P \\ & t \geq 0. \end{aligned} \tag{3.7}$$

The quadratic dependence on the variable t is undesirable and can be removed by changing variables to $r := t^2$, which yields

$$\begin{aligned} \underset{W,r,P \succ 0}{\text{minimize}} \quad & r \\ \text{subject to} \quad & W \in \mathcal{W} \\ & (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq r P \\ & r \geq 0. \end{aligned} \tag{3.8}$$

The next step is to work on the matrix inequality in (3.8) in order to induce a Schur complement form. By changing variables as $Q := P^{1/2}$, the matrix inequality can be represented as

$$\begin{aligned} (W - \mathrm{J})^T Q \frac{1}{r} Q (W - \mathrm{J}) \preceq Q^2 \Leftrightarrow \\ \Leftrightarrow Q^2 - (W - \mathrm{J})^T Q \frac{1}{r} Q (W - \mathrm{J}) \succ 0 \Leftrightarrow \\ \Leftrightarrow \begin{bmatrix} Q^2 & (W - \mathrm{J})^T Q \\ Q(W - \mathrm{J}) & rI \end{bmatrix} \succeq 0, \end{aligned} \tag{3.9}$$

where the last identity comes from the Schur's complement identity, a well-know technique in semidefinite programming.

Except for the term Q^2, condition (3.9) is convex in (Q, r) (with W fixed) and (W, r) (with Q fixed). This suggests the possibility of an alternate minimization approach in which it is possible to optimize with respect to (Q, r) and then (W, r), repeatedly. To make this approach viable, the term Q^2 was linearized. More precisely, since an update on Q can be viewed as an additive perturbation by a symmetric matrix Δ, i.e., $Q_k = Q_{k-1} + \Delta$, resulting in

$$(Q_{k-1} + \Delta)^2 = Q_{k-1}^T Q_{k-1} + Q_{k-1}^T \Delta + \Delta^T Q_{k-1} + \Delta^T \Delta \tag{3.10}$$

$$\approx Q_{k-1}^T Q_{k-1} + Q_{k-1}^T \Delta + \Delta^T Q_{k-1} \tag{3.11}$$

where the quadratic terms on Δ have been discarded. This makes the expression affine in Δ, which is the variable to optimize over. Note that since $\Delta^T \Delta \succeq 0$, the affine expression is, in fact, a matrix lower bound on $(Q_{k-1} + \Delta)^T (Q_{k-1} + \Delta)$; thus, by passing to the affine expression, the original feasible set is being restricted.

The steps above give a hint to a possible algorithm. This algorithm has the following steps:

- Find a starting W matrix and a starting Q_0 matrix;
- Fix W and Q_0 and solve the problem to find Δ. Update $Q_1 \leftarrow Q_0 + \Delta$;
- Fix Q_1 and find a new W matrix, and repeat.

Finally, the algorithm to minimize the spectral radius is the following:

1) Initialise $W_0 = \arg \min_{W \in \mathcal{W}} ||W - \mathrm{J}\,||$ and Q_0 according to:

$$\begin{aligned} \underset{k,P}{\text{minimize}} \quad & k \\ \text{subject to} \quad & I_n \leq P \leq k I_n \\ & (W_0 - J)^T P (W_0 - J) \leq (\rho_0 + \epsilon)^2 P. \end{aligned} \tag{3.12}$$

2) Solve:

$$\begin{array}{ll} \underset{r,\Delta}{\text{minimize}} & r \\ \text{subject to} & \Delta = \Delta^T \\ & Q_{k-1} + \Delta \succeq 0 \\ & \begin{bmatrix} Q_{k-1}^2 + \Delta Q_{k-1} + Q_{k-1}\Delta & (W_{k-1} - J)^T(Q_{k-1} + \Delta) \\ (Q_{k-1} + \Delta)^T(W_{k-1} - J) & rI_n \end{bmatrix} \succeq 0; \\ & \|\Delta\|_{\mathsf{F}} \leq \alpha \, \|Q_{k-1}\|_{\mathsf{F}} \end{array} \tag{3.13}$$

3) Set: $Q_k = Q_{k-1} + \Delta^*$

4) Solve:

$$\begin{array}{ll} \underset{r,W}{\text{minimize}} & r \\ \text{subject to} & \begin{bmatrix} Q_k^2 & (W - J)^T Q_k \\ Q_k^T(W - J) & rI_n \end{bmatrix} \succeq 0 \\ & 1^T W = 1^T \\ & W1 = 1 \\ & W \in \mathcal{S}; \end{array} \tag{3.14}$$

5) Set: $W_k = W^*$;

6) Repeat 2) through 5) until $||W_k - W_{k-1}||/||W_{k-1}|| \leq 1\%$.

Results. The P-Q algorithm was applied in five different ways: without any control of the norm of Δ, with Δ's norm constrained to 1% of $||Q_{k-1}||_{Fro}$, and also 2%, 10% and 20%. Figure 3.2 compares the results obtained with the five methods.

The unconstrained method improves the initial result by 4.2%, but stops after 5 iterations. This is a good improvement when compared to the previous results obtained. Since the P-Q algorithm suffered a linearization where a small Δ was introduced, the norm constraints to Δ were inserted, to see if the results would improve. There was a clear benefit in constraining Δ, but due to the different results obtained it is possible to state that the best norm constraint is an optimization problem in itself.

Finally, it is more important to state that this method is a step forward in the problem of optimizing the spectral radius as all tests made clearly reduced its value, to a minimum of 0.7268 obtained by the method with 2% norm constraint.

3.4 Successive Linearizations Algorithm

The Successive Linearizations Algorithm shows another way to think about the spectral radius minimization problem. The basis for this algorithm is the property of the spectral radius that was used in section 3.3, given by equation (3.6):

$$\exists_{P \succ 0} : (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq r^2 P.$$

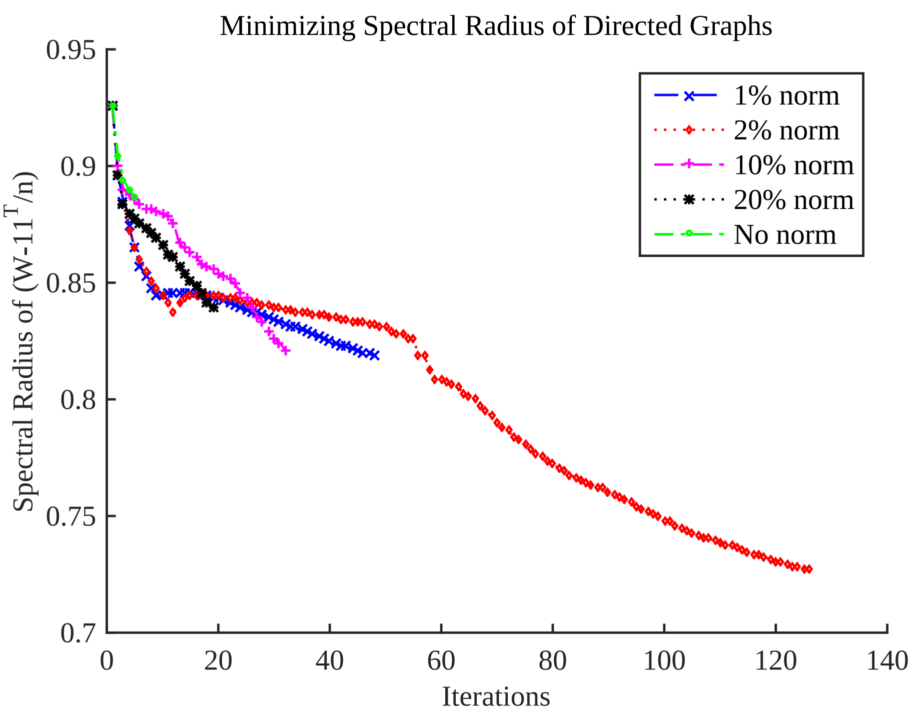


Figure 3.2: P-Q algorithm results.

This algorithm, just like the one described in section 3.3, initializes both W_0 and r as

$$W_0 = \arg\min_{W \in \mathcal{W}} ||W - \mathrm{J}\,||_2$$

and $r = \rho(W_0 - \mathrm{J})$. Since these variables are fixed, it is possible to search for a matrix P that obeys equation (3.6). From a numerical viewpoint, it is convenient to search for a well-conditioned $P \succ 0$, i.e., with the smallest condition number

$$\mathrm{cond}(P) = \frac{\lambda_{\mathrm{max}}(P)}{\lambda_{\mathrm{min}}(P)}. \tag{3.15}$$

That is, the optimization problem to solve:

$$\begin{array}{ll} \underset{P}{\text{minimize}} & \mathrm{cond}(P) \\ \text{subject to} & P \succ 0 \\ & (W_0 - J)^T P (W_0 - J) \leq (r + \epsilon)^2 P, \end{array} \tag{3.16}$$

where $\epsilon \geq 0$ is added to guarantee numerically that a matrix P can be found. As it stands, this optimization problem is not convex; in particular, it cannot be formulated as a SDP. To transform it into a convex problem, some reformulations are needed. The first step is to apply the epigraph method to obtain:

$$\begin{array}{ll} \underset{k,P}{\text{minimize}} & k \\ \text{subject to} & P \succ 0 \\ & \mathrm{cond}(P) \leq k \\ & (W_0 - J)^T P (W_0 - J) \leq (r + \epsilon)^2 P. \end{array} \tag{3.17}$$

The second step is to work the second constraint, cond$(P) \leq k$, so that it fits in the SDP model. To begin, substitute cond(P) by its formula:

$$\begin{aligned} \text{cond}(P) \leq k &\Leftrightarrow \frac{\lambda_{\text{max}}(P)}{\lambda_{\text{min}}(P)} \leq k \\ &\Leftrightarrow \lambda_{\text{max}}(P) \leq k\lambda_{\text{min}}(P) \end{aligned} \tag{3.18}$$

Now, it is possible to apply the fact that $\lambda_{\text{max}}(P) \leq k\lambda_{\text{min}}(P) \Leftrightarrow P \preceq k\lambda_{\text{min}}(P)I_n$ and change (3.18) into a form that is closer to an SDP-like constraint. Also notice that P is Definite Positive, $P \succ 0$, and all its eigenvalues $\lambda_i \geq \lambda_{\text{min}}(P) > 0$ for $i = 2, \ldots, n$. This can be expressed as $P \succeq \lambda_{\text{min}}(P)I_n$. Assembling these facts yields:

$$\lambda_{\text{min}}(P)I_n \preceq P \preceq k\lambda_{\text{min}}(P)I_n. \tag{3.19}$$

Constraint (3.19) is still not convex, but that problem can be solved with a change of variable $\bar{P} := P/\lambda_{\text{min}}(P)$, which is well-defined because $\lambda_{\text{min}}(P) > 0$.

Thus the final problem is:

$$\begin{aligned} \underset{k,P}{\text{minimize}} \quad & k \\ \text{subject to} \quad & P \succ 0 \\ & I_n \preceq P \preceq kI_n \\ & (W_0 - J)^T P (W_0 - J) \leq (r + \epsilon)^2 P. \end{aligned} \tag{3.20}$$

At this stage, W_0, r, and P_0 are determined and the struggle is to improve the spectral radius of W, i.e., to find new W and P that solve $(W - J)^T\ P\ (W - J) \leq \alpha^2\ P$, where $\alpha < r$, is the new objective. This constraint is not convex since, for example, W^TPW involves a trilinear interaction of variables.

Resorting to the previously used technique of fixing Q and trying to find a Δ_Q that shifts the matrices, it is possible to linearize the problem. However, these Δ's and the resulting matrices $Q+\Delta_Q$ and $P+\Delta_P$ need to respect the constraints set before. To linearize the problem, both Δ's need to be small shifts, $||\Delta_P|| \ll 1$ and $||\Delta_Q|| \ll 1$, and the resulting matrices need to follow $P + \Delta_P \succ 0$ and $W + \Delta_W \in \mathcal{W}$. With these modifications, the equation in (3.6) changes into:

$$((W + \Delta_W) - J)^T\ (P + \Delta_P)\ ((W + \Delta_W) - J) \leq r\ (P + \Delta_P). \tag{3.21}$$

Eliminating the second order terms, because the Δ's are small, the final resulting equation is:

$$\begin{aligned} &(W_k - J)^T\ P_k\ (W_k - J) + \\ &+ \Delta_W^T\ P_k\ (W_k - J) + (W_k - J)^T\ P_k\ \Delta_W + \\ &(W_k - J)^T\ \Delta_P\ (W_k - J) \leq \alpha^2\ (P_k + \Delta_P). \end{aligned} \tag{3.22}$$

This problem is posed in a way that hints to a feasibility problem: to find the Δ's that fit the constraints, without trying to minimize any objective function. This happens because, with $\alpha < r$, the problem is already being solved for a matrix W_k that has a better performance than the starting one, W_{k-1}. Once matrix W_k is found, a new more stable matrix P can be calculated, and the process can be repeated.

According to the steps described the algorithm has this form:

1) Initialise $W_0 = \arg\min_{W \in \mathcal{W}} ||W - \mathrm{J}\,||$, $\epsilon = 10^{-6}$, and $\rho_0 = \rho(W_0 - J)$;

2) Solve:

$$\begin{array}{ll} \underset{k,P}{\text{minimize}} & k \\ \text{subject to} & I_n \leq P \leq kI_n \\ & (W_0 - J)^T P (W_0 - J) \leq (\rho_0 + \epsilon)^2 P; \end{array} \tag{3.23}$$

3) Solve the feasibility problem for Δ_W and Δ_P subject to:

$$\begin{array}{l} \text{i) } P_k + \Delta_P > 0 \\ \text{ii) } W_k + \Delta_W \in \mathcal{W} \\ \text{iii) } (W_k - J)^T\ P_k\ (W_k - J) + \\ \quad + \Delta_W^T\ P_k\ (W_k - J) + (W_k - J)^T\ P_k\ \Delta_W + \\ \quad (W_k - J)^T\ \Delta_P\ (W_k - J) \leq [0.99(\rho_k + \epsilon)]^2\ (P_k + \Delta_P) \\ \text{iv) } ||\Delta_W||_F \leq 0.1\ ||W_k||_\mathrm{F} \\ \text{v) } ||\Delta_P||_F \leq 0.1\ ||P_k||_\mathrm{F}; \end{array} \tag{3.24}$$

4) Update $\rho_{k+1} = 0.99\rho_k$ and $W_{k+1} = W_k + \Delta_W$;

5) Check if the algorithm failed to obtain a better spectral radius, $\rho(W_{k+1} - J) \leq \rho_{k+1}$.

6) If $\rho(W_{k+1} - J) \leq \rho_{k+1}$ is true, update P according to method (3.25) and repeat 3) to 6).

$$\begin{array}{ll} \underset{k,P}{\text{minimize}} & k \\ \text{subject to} & I_n \leq P \leq kI_n \\ & (W_{k+1} - J)^T P (W_{k+1} - J) \leq (\rho_{k+1} + \epsilon)^2 P; \end{array} \tag{3.25}$$

Results. Figure 3.3 shows the results obtained from the Successive Linearizations Algorithm. The method performs 4 iterations and achieves a final spectral radius of $\rho(W - \mathrm{J}) = 0.8988$. This result increases the performance in 2.9%, thus performing worse than the P-Q algorithm.

3.5 Cool Down

Note: After the thesis was defended, the authors learned that reference [10] proposed an almost identical algorithm to the Cool Down algorithm that was developed in Section 3.5.

In this section a new approach is developed based on equation (3.6):

$$\rho\,(W - \mathrm{J}) \leq r \quad \Leftrightarrow \quad \exists_{P \succ 0} : (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq r^2 P.$$

The approach consists in fixing the variable r to a target value and searching for matrices P and W that make the inequality true: if such matrices can be found, the spectral radius of $W - \mathrm{J}$ has been certified to be less or equal to r. Then r is decreased by a small amount and repeat the process.

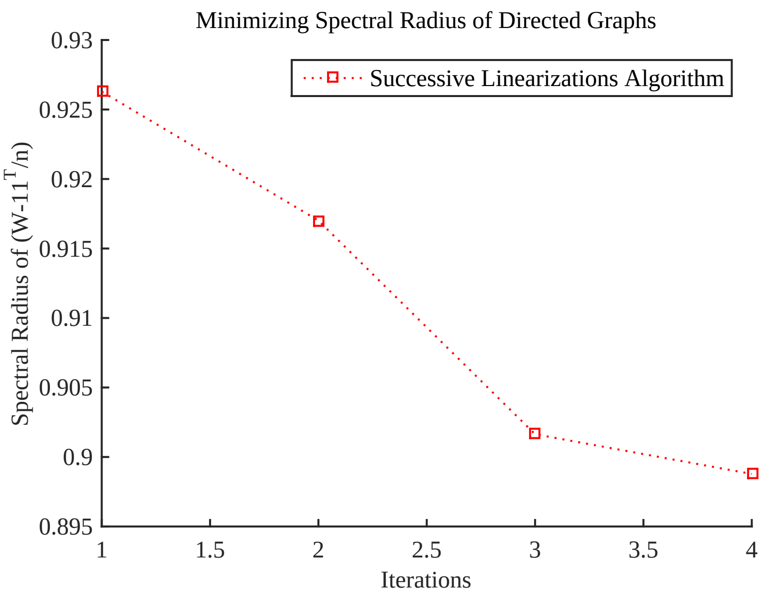


Figure 3.3: Successive Linearizations algorithm results.

At each step, the faced feasibility problem is:

$$\begin{aligned} &\text{Find} && P \text{ and } W \\ &\text{subject to} && P \succ 0 \qquad W \in \mathcal{W} \\ & && (W - J)^T P (W - J) \leq r^2 P. \end{aligned} \tag{3.26}$$

The problem is not convex as explained in section 3.4 but a couple of techniques can be applied to transform it into an SDP. Step one involves adding a new variable $Q \succ 0$, where $P = Q^{-1}$, such that Schur's Complement can be applied,

$$\begin{aligned} (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq r^2 P \quad &\Leftrightarrow \quad (W - \mathrm{J})^T Q^{-1} (W - \mathrm{J}) \preceq r^2 P \\ &\Leftrightarrow \quad r^2 P - (W - \mathrm{J})^T Q^{-1} (W - \mathrm{J}) \succeq 0, \end{aligned} \tag{3.27}$$

resulting in

$$\begin{aligned} &\text{Find} && P,\ W \text{ and } Q \\ &\text{subject to} && P \succ 0 \qquad Q \succ 0 \\ & && P = Q^{-1} \qquad W \in \mathcal{W} \\ & && \begin{bmatrix} r^2 P & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q \end{bmatrix} \succeq 0. \end{aligned} \tag{3.28}$$

Since

$$P = Q^{-1} \Leftrightarrow \begin{cases} P \succeq Q^{-1} \\ \mathrm{Tr}(PQ) = n \end{cases}$$

equation (3.28) can be reformulated as

$$\begin{aligned}
&\text{Find} && P,\ W \text{ and } Q \\
&\text{subject to} && P \succ 0 \qquad Q \succ 0 \\
& && P \succeq Q^{-1} \qquad \mathrm{Tr}(PQ) = n \\
& && W \in \mathcal{W} \\
& && \begin{bmatrix} r^2 P & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q \end{bmatrix} \succeq 0.
\end{aligned} \tag{3.29}$$

Except for the constraint $\mathrm{Tr}(PQ) = n$, this is a SDP feasibility problem. To handle the constraint, it is transformed into an objective:

$$\begin{aligned}
&\underset{P,\ Q,\ W}{\text{minimize}} && \mathrm{Tr}(PQ) \\
&\text{subject to} && P \succ 0 \qquad Q \succ 0 \\
& && W \in \mathcal{W} \\
& && \begin{bmatrix} P & I \\ I & Q \end{bmatrix} \succeq 0 \\
& && \begin{bmatrix} r^2 P & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q \end{bmatrix} \succeq 0.
\end{aligned} \tag{3.30}$$

Thus, the goal becomes the minimization of $\mathrm{Tr}(PQ)$ over the remaining set of constraints; note that, for (P, Q, W) feasible in (3.30), there holds $\mathrm{Tr}(PQ) \geq n$ since the constraint $P \succeq Q^{-1}$ implies $Q^{1/2} P Q^{1/2} \succeq I$, and therefore $\mathrm{Tr}(Q^{1/2} P Q^{1/2}) = \mathrm{Tr}(PQ) \ \geq\ n$. This means that if (3.30) is solved, the remaining test is to check if $\mathrm{Tr}(PQ) = n$ holds at the solution. If so, the problem (3.29) is feasible and the solution W from (3.30) satisfies $\rho\,(W - J) \leq r$.

The final step is to linearize the objective $\mathrm{Tr}(PQ)$ since it is not convex. By introducing symmetric variables Δ_P and Δ_Q the problem to solve becomes a SDP with the following formulation:

$$\begin{aligned}
&\underset{W, \Delta_P, \Delta_Q}{\text{minimize}} && \mathrm{Tr}[(P + \Delta_P)\, Q + P\, (Q + \Delta_Q)] \\
&\text{subject to} && \begin{bmatrix} r^2 (P + \Delta_P) & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q + \Delta_Q \end{bmatrix} \succeq 0 \\
& && \begin{bmatrix} P + \Delta_P & I_n \\ I_n & Q + \Delta_Q \end{bmatrix} \succeq 0 \\
& && W \in \mathcal{W}
\end{aligned} \tag{3.31}$$

and the final algorithm is

1) Initialise $W_0 = \arg \min\limits_{W \in \mathcal{W}} ||W - \mathrm{J}\,||$, $\epsilon = 10^{-6}$, and $\eta = \rho(W_0 - J)$;

2) Find P_0 by solving:

$$\begin{aligned}
&\underset{k, P}{\text{minimize}} && k \\
&\text{subject to} && I_n \leq P \leq k I_n \\
& && (W_0 - J)^T P (W_0 - J) \leq (\eta^2 + \epsilon) P;
\end{aligned} \tag{3.32}$$

3) Update variables $\eta = 0.95\eta$ and $P = P_0$, and initialise $Q = P_0^{-1}$;

4) Solve the following iterative algorithm:

$$\begin{aligned}
\underset{W,\Delta_P,\Delta_Q}{\text{minimize}} \quad & \mathrm{Tr}[(P+\Delta_P)\,Q + P\,(Q+\Delta_Q)] \\
\text{subject to} \quad & \begin{bmatrix} \eta^2(P+\Delta_P) & (W-J)^T \\ (W-J) & Q+\Delta_Q \end{bmatrix} \succeq 0 \\
& \begin{bmatrix} P+\Delta_P & I_n \\ I_n & Q+\Delta_Q \end{bmatrix} \succeq 0 \\
& 1^T W = 1^T \\
& W1 = 1 \\
& W \in \mathcal{S};
\end{aligned} \tag{3.33}$$

5) Update variables $P = P + \Delta_P$ and $Q = Q + \Delta_Q$. Repeat 4) and 5) until:

$$\frac{||\Delta_P||_F}{||P||_F} \leq 0.1 \quad \frac{||\Delta_Q||_F}{||Q||_F} \leq 0.1; \tag{3.34}$$

6) Evaluate the stopping criteria $\mathrm{Tr}(PQ) \leq n + \epsilon$. If the condition is met update η:

$$\eta = 0.95\eta; \tag{3.35}$$

and repeat 4) through to 6). Otherwise, terminate the algorithm.

Results. This algorithm achieved very good results, as shown in Figure 3.4. As the algorithm solves for lower spectral radius values, it takes increasingly more time, but the boost in performance is noticeable. This algorithm achieved an increase of $46,6\%$ in performance for the directed graph previously used, and to verify the results are correct, a new graph was randomly generated for which this method obtained a $56,4\%$ increase in performance! These performance increases are calculated with the value obtained in the lowest point of the plot. The end values of the plot increase the spectral radius because no stopping criteria was defined, and the algorithm was looking for a spectral radius that it couldn't reach. The algorithm was stopped when the user observed the increasing values of the spectral radius.

3.6 Penalty based approaches

This section explores a further set of reformulations to directly minimize r in equation (3.6). Notice that, in the previous Cool Down approach, the upper-bound r was not an optimization variable—it was fixed at sequentially decreasing target values. In this approach, it is added to the set of variables.

Start from the non-convex formulation of the problem:

$$\begin{aligned}
\underset{r,P,W}{\text{minimize}} \quad & r \\
\text{subject to} \quad & W \in \mathcal{W} \quad P \succ 0 \quad r > 0 \\
& (W - \mathrm{J})^T\, P\, (W - \mathrm{J}) \preceq r^2 P.
\end{aligned} \tag{3.36}$$

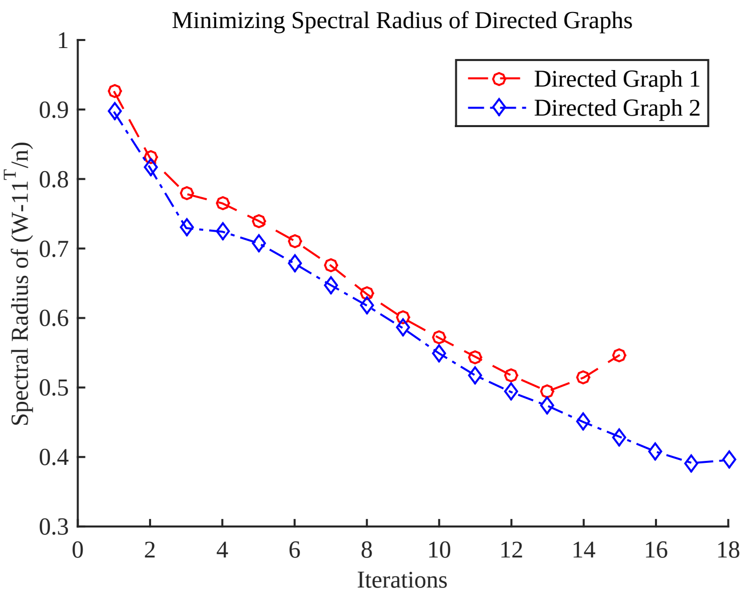


Figure 3.4: Cool Down algorithm results.

The problem needs to be transformed into a convex one, and the first step is to change the constraint with the r^2P term into a more manageable form:

$$\begin{aligned} \underset{r,P,W}{\text{minimize}} \quad & r \\ \text{subject to} \quad & W \in \mathcal{W} \quad P \succ 0 \quad r > 0 \\ & \frac{(W - \mathrm{J})^T}{r}\, P\, \frac{(W - \mathrm{J})}{r} \preceq P. \end{aligned} \tag{3.37}$$

By changing variables as $t = 1/r$ and $\bar{W} = t\,W$, the objective function also changes from r to $-t$, resulting in

$$\begin{aligned} \underset{t,P,\bar{W}}{\text{minimize}} \quad & -t \\ \text{subject to} \quad & \bar{W} \in \bar{\mathcal{W}} \quad P \succ 0 \quad t > 0 \\ & (\bar{W} - t\,\mathrm{J})^T\, P\, (\bar{W} - t\,\mathrm{J}) \leq P. \end{aligned} \tag{3.38}$$

Here, $\bar{\mathcal{W}}$ is the set of constraints $\bar{W}\mathbf{1} = t\mathbf{1}$, $\mathbf{1}^T\bar{W} = t\mathbf{1}^T$ and $\bar{W} \in \mathcal{S}$.

Finally, introduce a new variable, $Q = P^{-1}$, to turn the non-convex matrix inequality into a convex one:

$$\begin{aligned} \underset{t,P,\bar{W},Q}{\text{minimize}} \quad & -t \\ \text{subject to} \quad & \bar{W} \in \bar{\mathcal{W}} \quad P \succ 0 \quad t > 0 \\ & (\bar{W} - t\,\mathrm{J})^T\, Q^{-1}\, (\bar{W} - t\,\mathrm{J}) \preceq P \\ & Q = P^{-1}. \end{aligned} \tag{3.39}$$

At this point, the non-convexity of the problem is concentrated in the last constraint, $Q = P^{-1}$. In the following, two techniques are proposed to handle it: a majorization-minimization and a distance from Identity matrix approach.

Majorization-minimization technique. The non-convex matrix constraint $Q = P^{-1}$ is equivalent to a convex matrix inequality and a scalar non-convex equality:

$$Q = P^{-1} \Leftrightarrow \begin{cases} Q \succeq P^{-1} \\ \operatorname{Tr}(Q - P^{-1}) = 0. \end{cases} \tag{3.40}$$

This fact can be used by letting the convex matrix inequality stay as a constraint and trying to enforce the scalar inequality by penalizing it in the objective:

$$\begin{array}{ll} \underset{r,P,W}{\text{minimize}} & -t + \beta \operatorname{Tr}(Q - P^{-1}) \\ \text{subject to} & \bar{W} \in \bar{\mathcal{W}} \quad P \succeq 0 \quad t > 0 \\ & (\bar{W} - t\,\mathrm{J})^T\, Q^{-1}\, (\bar{W} - t\,\mathrm{J}) \leq P \\ & Q \succeq P^{-1}, \end{array} \tag{3.41}$$

where $\beta > 0$ is a constant.

Problem (3.41) is not convex since the objective function is not convex. However, the objective has a valuable structure: it is the difference of two convex functions. Indeed:

$$-t + \beta \operatorname{Tr}(Q - P^{-1}) = \overbrace{\underbrace{-t + \beta \operatorname{Tr}(Q)}_{\text{convex}} - \underbrace{\operatorname{Tr}(P^{-1})}_{\text{convex}}}^{\text{not convex}}. \tag{3.42}$$

This structure can be exploited for constructing an iterative algorithm that, by solving a series of convex problems, sequentially minimizes the objective function of (3.41). This approach is generically known as majorization-minimization or convex-concave procedure. For problems with this structure, a brief explanation is now given on how the method works. Begin with

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) - g(x) \\ \text{subject to} & x \in \mathcal{X}, \end{array} \tag{3.43}$$

where both $f(x)$ and $g(x)$ are convex (and $f(x) - g(x)$ is not) and $\mathcal{X}$ is a convex set. Let $x_k \in \mathcal{X}$ and linearize g at x_k: it is well known that this first-orderTaylor expansion produces a lower-bound for $g(x)$, i.e.,

$$\forall_x : \; g(x) \geq g(x_k) + \nabla g(x_k)^T (x - x_k), \tag{3.44}$$

which leads to

$$f(x) - g(x) \leq f(x) - g(x_k) - \nabla g(x_k)^T (x - x_k). \tag{3.45}$$

The function on the right-hand side of (3.45) is convex, coincides with the true objective function $f(x) - g(x)$ at the point $x = x_k$, and dominates it everywhere else. Thus, by minimizing this convex upper-bound, progress is made indirectly on the objective of interest, $f - g$. Indeed, letting

$$x_{k+1} := \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, f(x) - \nabla g(x_k)^T (x - x_k) \tag{3.46}$$

note that

$$\begin{aligned} f(x_{k+1}) - g(x_{k+1}) &\leq f(x_{k+1}) - g(x_k) - \nabla g(x_k)^T (x_{k+1} - x_k) \\ &\leq f(x_k) - g(x_k) - \nabla g(x_k)^T (x_k - x_k) \\ &= f(x_k) - g(x_k). \end{aligned}$$

The first inequality follows from (3.45) applied at $x = x_{k+1}$. The second inequality follows from the definition of x_{k+1} in (3.46). Note that, although progress is made in passing from x_k to x_{k+1}, the sequence $(x_k)_{k\geq 1}$ is not guaranteed to converge to a global minimum of problem (3.43).

To apply this methodology to the objective function in (3.42), a linearization of the term $\mathrm{Tr}(P^{-1})$ at a generic point $P_k \succ 0$ is needed. This computation is well known in linear algebra and corresponds to

$$\mathrm{Tr}(P^{-1}) \geq \mathrm{Tr}(P_k^{-1}) - \ \mathrm{Tr}(P_k^{-1}(P - P_k)P_k^{-1}). \tag{3.47}$$

The final step is to assemble (3.42) and (3.47), and complete the formulation of the problem to solve, which is

$$\begin{aligned} \underset{t,P,\bar{W},Q}{\text{minimize}} \quad & -t + \beta\,\mathrm{Tr}(Q) + \beta\ \mathrm{Tr}(P_k^{-1}\ P\ P_k^{-1})) \\ \text{subject to} \quad & \bar{W} \in \bar{\mathcal{W}} \quad P \succ 0 \quad t > 0 \\ & (\bar{W} - t\,\mathrm{J})^T\ Q^{-1}\ (\bar{W} - t\,\mathrm{J}) \preceq P \\ & Q \succeq P^{-1}. \end{aligned} \tag{3.48}$$

Two algorithms are now proposed to deal with this formulation. The algorithms differ mainly in the choice of the first P_0. In the first algorithm, the normalization $\mathrm{Tr}(P_0) = 1$ is enforced and kept throughout the following iterations; in the second algorithm, a P_0 with minimal condition number is computed and no further normalizations are enforced.

Algorithm 1. Algorithm 1 is laid out below with a simple explanation for its steps.

1) Initialise $W_0 = \arg\min_{W\in\mathcal{W}} ||W - \mathrm{J}\,||$, $\epsilon = 10^{-6}$, and $\eta = \rho(W_0 - J)$;

2) Find P_0 by solving a feasibility problem based on the following constraints:

$$\begin{aligned} &\text{i) } P_0 \succeq 0 \\ &\text{ii) } (W_0 - J)^T\ P_0\ (W_0 - J) \leq (\eta + \epsilon)^2\ P_0 \\ &\text{iii) } \mathrm{Tr}(P_0) = 1 \end{aligned} \tag{3.49}$$

3) Solve the following minimization problem k times:

$$\begin{aligned} \underset{t,P}{\text{minimize}} \quad & -t + \beta\,\mathrm{Tr}(Q) + \beta\,\mathrm{Tr}(P_{k-1}^{-1} P P_{k-1}^{-1}) \\ \text{subject to} \quad & P \succeq 0; \\ & t > 0; \\ & \mathrm{Tr}(P) = 1 \\ & \begin{bmatrix} Q & I_n \\ I_n & P \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P & (\bar{W} - tJ)^T \\ (\bar{W} - tJ) & Q \end{bmatrix} \succeq 0 \\ & \bar{W} \in \bar{\mathcal{W}}; \end{aligned} \tag{3.50}$$

Algorithm 2. Algorithm 2 is laid out below with a simple explanation for its steps.

1) Initialise $W_0 = \arg\min_{W \in \mathcal{W}} ||W - \mathrm{J}\,||$, $\epsilon = 10^{-6}$, and $\eta = \rho(W_0 - J)$;

2) Find P_0 by solving:

$$\begin{aligned} \underset{k,P}{\text{minimize}} \quad & k \\ \text{subject to} \quad & I_n \leq P \leq kI_n \\ & (W_0 - J)^T P (W_0 - J) \leq (\eta^2 + \epsilon) P; \end{aligned} \tag{3.51}$$

3) Solve the following minimization problem k times:

$$\begin{aligned} \underset{t,P}{\text{minimize}} \quad & -t + \beta \operatorname{Tr}(Q) + \beta \operatorname{Tr}[P_{k-1}^{-1} P P_{k-1}^{-1}] \\ \text{subject to} \quad & P \succeq 0; \\ & t > 0; \\ & \begin{bmatrix} Q & I_n \\ I_n & P \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P & (\bar{W} - tJ)^T \\ (\bar{W} - tJ) & Q \end{bmatrix} \succeq 0 \\ & \bar{W} \in \bar{\mathcal{W}}. \end{aligned} \tag{3.52}$$

Distance from Identity matrix. Returning to the formulation (3.39), note that the troublesome constraint $Q = P^{-1}$ is equivalent to $QP = I_n$, which can be handled via a penalty approach:

$$\begin{aligned} \underset{t,P,Q,\bar{W}}{\text{minimize}} \quad & -t + \beta \, \|QP - I_n\|^2 \\ \text{subject to} \quad & P \succeq 0 \quad Q \succeq 0 \\ & \bar{W} \in \bar{\mathcal{W}} \quad t \geq 0 \\ & \begin{bmatrix} Q & I_n \\ I_n & P \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P & (\bar{W} - tJ)^T \\ (\bar{W} - tJ) & Q \end{bmatrix} \succeq 0. \end{aligned} \tag{3.53}$$

The formulation in (3.53) is amenable to alternate optimization: when P is fixed, the problem is convex with respect to the remaining variables; the same is true when Q is fixed. Thus, the following algorithm is suggested.

Algorithm 3.

1) Initialise $W_0 = \arg\min_{W \in \mathcal{W}} ||W - \mathrm{J}\,||$, $\epsilon = 10^{-6}$, and $\eta = \rho(W_0 - J)$;

2) Find P_0 by solving:

$$\begin{aligned} \underset{k,P}{\text{minimize}} \quad & k \\ \text{subject to} \quad & I_n \leq P \leq kI_n \\ & (W_0 - J)^T P (W_0 - J) \leq (\eta^2 + \epsilon) P; \end{aligned} \tag{3.54}$$

3) Find the optimal Q_k (P_{k-1} is fixed) by solving:

$$\begin{aligned} \underset{t,Q,\bar{W}}{\text{minimize}} \quad & -t+\beta||Q_kP_{k-1}-I_n||^2 \\ \text{subject to} \quad & Q_k \succeq 0 \\ & \bar{W}\in\bar{\mathcal{W}} \quad t\geq 0 \\ & \begin{bmatrix} Q_k & I_n \\ I_n & P_{k-1} \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P_{k-1} & (\bar{W}-tJ)^T \\ (\bar{W}-tJ) & Q_k \end{bmatrix} \succeq 0. \end{aligned} \tag{3.55}$$

4) Find the optimal P_k (Q_k is fixed) by solving:

$$\begin{aligned} \underset{t,P,\bar{W}}{\text{minimize}} \quad & -t+\beta||Q_kP_k-I_n||^2 \\ \text{subject to} \quad & P_k \succeq 0 \\ & \bar{W}\in\bar{\mathcal{W}} \quad t\geq 0 \\ & \begin{bmatrix} Q_k & I_n \\ I_n & P_k \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P_k & (\bar{W}-tJ)^T \\ (\bar{W}-tJ) & Q_k \end{bmatrix} \succeq 0. \end{aligned} \tag{3.56}$$

5) Repeat steps 3) and 4) until the iterations stagnate.

Numerical Results. The first three figures below represent the results obtained for each of the algorithms with different choice of the penalization coefficient β, and the fourth figure compares the best of each algorithm.

Notice in Figure 3.5 the difference in performance between the three curves. This means that the curve with $\beta = 1$ is closer to the optimal β that achieves the best performance.

Algorithm 2, shown in Figure 3.6, has an unexpected behaviour. The idea behind every algorithm is that the spectral radius will continuously decrease, and the curves show some unexpected oscillations. The stopping criterion used for this algorithm was 200 iterations.

Figure 3.7 shows the numerical results of Algorithm 3. Though it was expected that the spectral radius would decrease, as it does in the first iteration, it was surprising that it remained constant throughout the following iterations. Furthermore, this unexpected behaviour persisted for different values of β. The silver lining in this approach is that it increases the performance of the initial method, which is the aim of all algorithms.

Finally, comparing all three algorithms in Figure 3.8 reveals which one performed the best. In this case, it is evident to see that Algorithm 2 with $\beta = 5$ increases performance much better than the other two proposed algorithms. It is worth noting that even though this is by far the best algorithm, it still doesn't match the performance achieved by the Cool Down method.

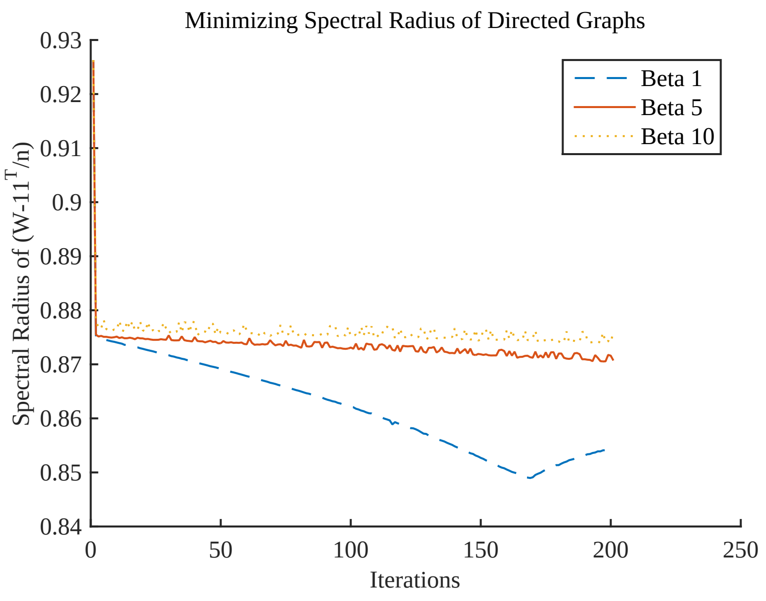


Figure 3.5: Algorithm 1 results.

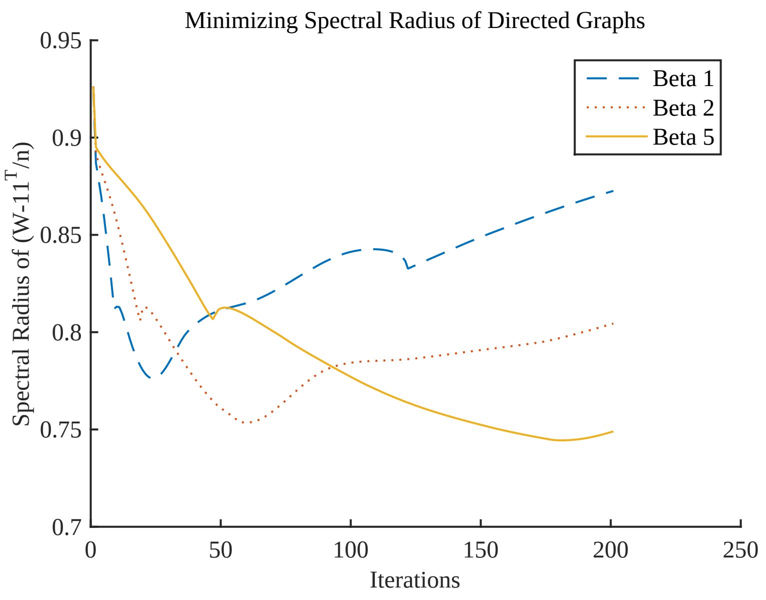


Figure 3.6: Algorithm 2 results.

3.7 Closing Remarks

In this chapter several approaches to minimizing the spectral radius of asymmetric networks were discussed. The first one used the spectral norm as a proxy for the spectral radius, and all others were convex problems generated from the well-known fact in equation (3.6). These methods differed in the quality of their results, and some of these formulations present parameters that could be further optimized.

The first method found a matrix W whose performance was clearly far away from the best performance possible, and not even a random search was able to improve that result significantly. The other

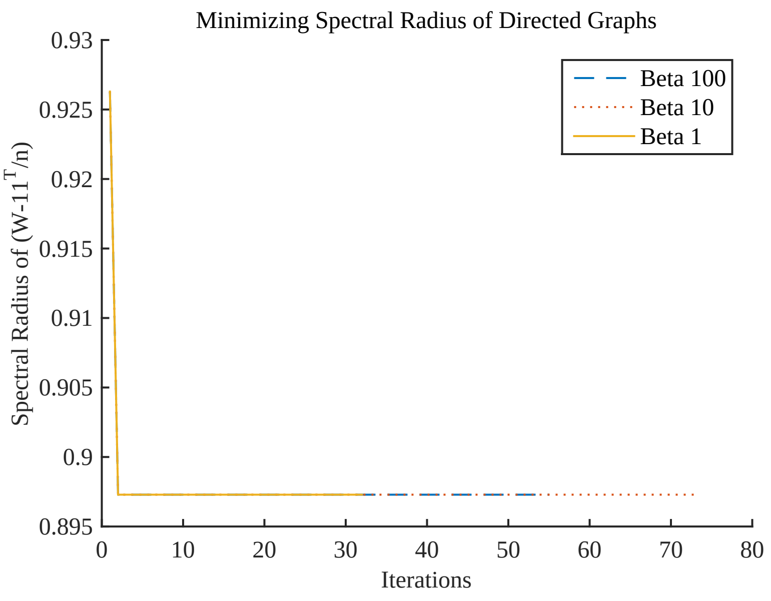


Figure 3.7: Algorithm 3 results.

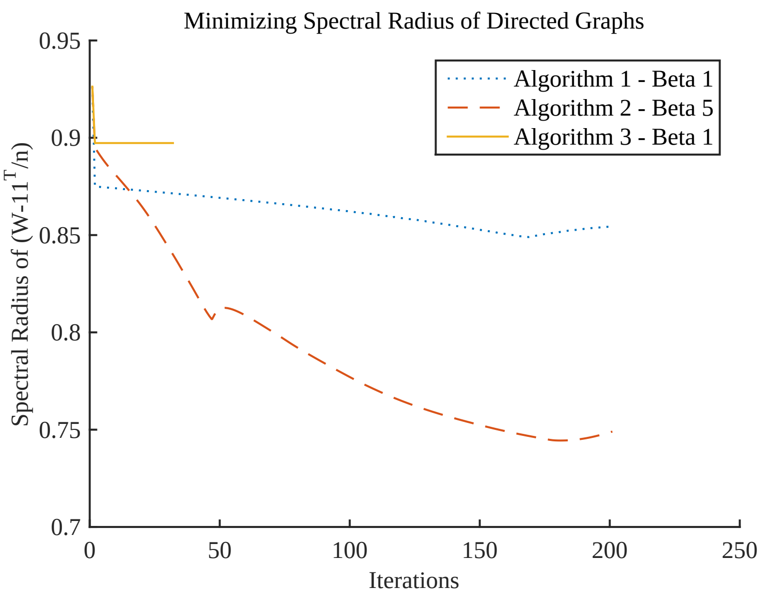


Figure 3.8: Comparison between the three algorithms.

methods were able to improve the result obtained via the spectral norm, with the Cool Down method achieving results that are very close to the optimum.

The results obtained from the Cool Down method were very interesting as the formulation presented appears to have solved the problem of minimizing the spectral radius. It is possible to state this because the best spectral radius of a matrix is a finite positive real number, and the Cool Down method approaches that limit as it successively sets a new spectral radius target. In this method it is possible to improve the stopping criterium and the step between each target that is set, in order to maximize the speed of convergence to the best solution.

It is important to talk about the speed of convergence of the Cool Down method because, from all the methods tested, this was the slowest and the time between each iteration increased significantly. This means that if the user is just looking to improve the performance of the weight matrix in a small percentage, it can resort to the other methods that were presented in this thesis.

Further exploration could be done in terms of finding out why it is that the formulation used in the Cool Down method presents results that are much better when compared to the other algorithms, even though they start from the same well known fact.

Chapter 4

Conclusions

This thesis developed novel approaches for the general consensus problem, in two directions.

The first part considered symmetric networks and proposed methods to make the networks substantially sparser without sacrificing too much their consensus speed. The second part considered asymmetric networks and proposed methods to accelerate consensus (without the aim of making the networks sparse).

The search for sparsity yielded several methods capable of significantly reducing the number of communication channels. This work developed efficient algorithms with complementary features. On one hand, the re-weighted ℓ_1-norm method provides a dramatic improvement in the network sparsity, with a negligible degradation in the consensus speed. On the other hand, the Heuristic used to grow graphs and the Dual methods offer a simple mechanism to add individual edges until a desired target number of channels is reached. Thus, in contrast with the re-weighted ℓ_1-norm method, they allow for a fine control on the final sparsity of the network.

In the second part, this work addressed a harder problem: the minimization of the spectral radius of an asymmetric matrix whose entries are constrained to a convex set. As the spectral radius of asymmetric matrices is a non-convex function of its entries (in sharp distinction with the situation for symmetric matrices), the resulting optimization problem is very challenging, which explains the fact that almost no work exists on this topic.

Throughout this work several methods were developed to tackle this problem, the most interesting of which was the Cool Down method. The Cool Down method improves significantly the networks obtained by using the spectral norm as a proxy, as well as the networks obtained by the other methods. It works by successively setting a target spectral radius to hit, which is carried out by solving a sequence of convex problems. Though it is an algorithm that takes a longer time to run, the increase in performance is pronounced.

Finally, further work could be done in optimizing each of the individual methods, as they all show parameters that can be tuned to find the best speed of convergence. Though the aim of this work was to find the matrix with the smallest spectral radius, the time it takes to design such a matrix may matter for certain applications, for example if a network is expected to work for 10 years, the matrix design could last 1 week without being a problem. Generically, the complexity of an algorithm to solve a SDP problem is n^6 and the algorithms developed throughout this work ran for up to five days (Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz - 64GB RAM), so this opens ground for future research, in the form of numerical refinement of the optimization methods developed in this thesis.

Bibliography

[1] J. N. Tsitsiklis, *Problems in decentralized decision making and computation*. PhD thesis, MIT, 1984.

[2] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” *Information Theory, IEEE Transactions on*, vol. 52, pp. 2508–2530, June 2006.

[3] B. Krishnamachari, D. Estrin, and S. Wicker, “The impact of data aggregation in wireless sensor networks,” in *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*, pp. 575–578, IEEE, 2002.

[4] I. Gupta, R. van Renesse, and K. Birman, “Scalable fault-tolerant aggregation in large process groups,” in *Dependable Systems and Networks, 2001. DSN 2001. International Conference on*, pp. 433–442, July 2001.

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, “Next century challenges: Scalable coordination in sensor networks,” in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom ’99, (New York, NY, USA), pp. 263–270, ACM, 1999.

[6] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “Tag: A tiny aggregation service for ad-hoc sensor networks,” *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, Dec. 2002.

[7] J. Fax and R. Murray, “Information flow and cooperative control of vehicle formations,” *Automatic Control, IEEE Transactions on*, vol. 49, pp. 1465–1476, Sept 2004.

[8] D. Kempe, J. Kleinberg, and A. Demers, “Spatial gossip and resource location protocols,” *J. ACM*, vol. 51, pp. 943–967, Nov. 2004.

[9] S. Aldosari and J. Moura, “Distributed detection in sensor networks: Connectivity graph and small world networks,” in *Signals, Systems and Computers, 2005. Conference Record of the Thirty-Ninth Asilomar Conference on*, pp. 230–234, Oct 2005.

[10] S. You, “A fast linear consensus protocol on an asymmetric directed graph,” in *2014 American Control Conference*, pp. 3281–3286, June 2014.

[11] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[12] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems &amp; Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[13] A. Ghosh and S. Boyd, “Growing well-connected graphs,” in *Decision and Control, 2006 45th IEEE Conference on*, pp. 6605–6611, Dec 2006.

[14] D. P. Bertsekas, *Rollout Algorithms for Discrete Optimization: A Survey*, pp. 2989–3013. Springer New York, 2013.