Multiagent System Optimization

José Miguel Filipe Antunes
jose.antunes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2016

Abstract

Consensus is a core tool in distributed multi-agent networks in which communication is very sparse, i.e., each agent can only communicate with a few neighbors. At time 0, each agent holds some private measurement; the goal of any consensus algorithm is to compute the average of those measurements, via successive exchange of messages between agents, and to make the average appear at each agent. Consensus algorithms use two key resources: communications (the number of channels used by the agents) and time (how long it takes for the average of the initial measurements to surface at each agent). Consensus algorithms thrive to be fast while using a modicum of communication channels.

In this thesis, we develop consensus algorithms that substantially improve the state-of-art in these two critical directions: communications and time. The first part of the thesis focuses on communications, while the second part focus on time.

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

In the second part of the thesis, we develop methods that accelerate the speed of convergence of consensus algorithms. We address this problem for the challenging setup of asymmetric networks, i.e., networks in which agents may not communicate bidirectionally (an agent can send messages to a neighbor agent, but not vice-versa). From the mathematical standpoint, the task translates into the minimization of the spectral radius of asymmetric matrices whose entries are constrained to a convex set—a notoriously difficult non-convex problem. 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%.

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

Note: After the thesis was defended, the authors learned that reference [5] proposed an almost identical algorithm to the Cool Down algorithm that was developed in Section 3.2.

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 $\mathcal{V}=\{1,\dots,n\}$ 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

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

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 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 $t=0,1,2,\dots$ 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 $i=1,\dots,n$, and the states are updated as:

$$x_i^{(t+1)} = W_{ii}x_i^{(t)} + \sum_{j\in N_i} W_{ij}x_j^{(t)} \quad ,i = 1,\dots,n. \quad (2)$$

Equation (2) shows that each agent repeatedly performs a weighted average of its estimate with its neighbours' estimates. Equation (2) can be rewritten in a compact vector notation:

$$x^{(t+1)} = Wx^{(t)} \quad (3)$$

where $x^{(t)} := (x_1^{(t)},\dots,x_n^{(t)}) \in \mathbf{R}^n$ 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.

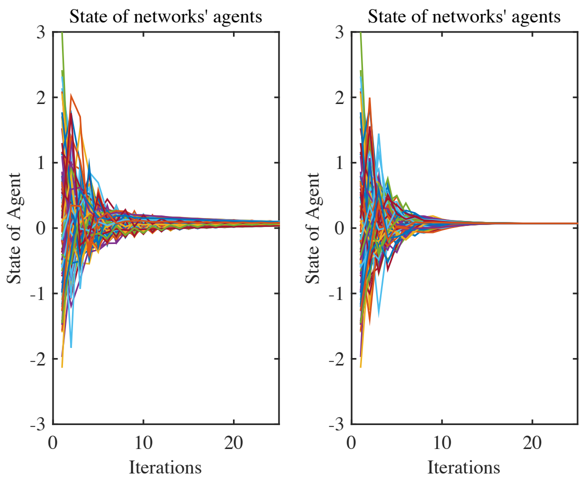
The design of W is a critical issue as not all weight matrices lead to consensus [4], that is, not all matrices W ensure that

$$x_i^{(t)} \to \overline{x} \quad \text{as} \quad t\to\infty \quad ,\text{for all} \quad i = 1,\dots,n. \quad (4)$$

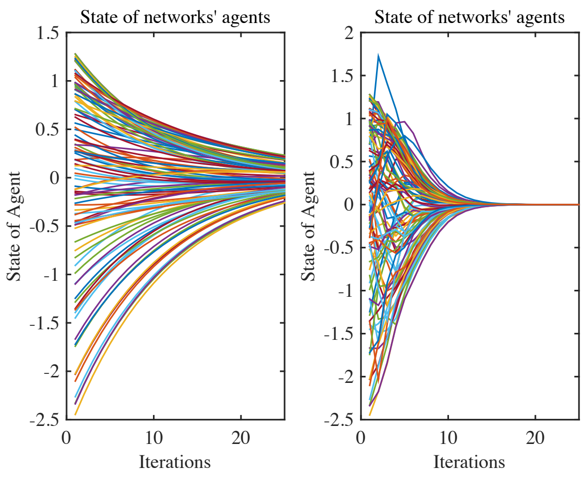
Preview of the main achievements. The first part of this thesis focused on designing matrices W that secure (4) and are sparse, i.e., the underlying distributed algorithm (3) uses few communication channels. In the second part of this thesis, the focus was on designing asymmetric matrices W that yield fast convergence rates of the iterative scheme (3). Since the topic addressed in the second part of the thesis is relatively new territory, the following paragraphs offer a sample of the impact of this works' achievements in three important applications: standard consensus, event detection, and target tracking. It also compares the state-of-art method (which this work calls Naive method) with the best method obtained(which is called Cool Down and explained in section 3.2.)

Standard consensus. Observe figures 1a and 1b. Figure 1a 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 1b. 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.

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 sooner. Look at figures 2a and 2b. 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.

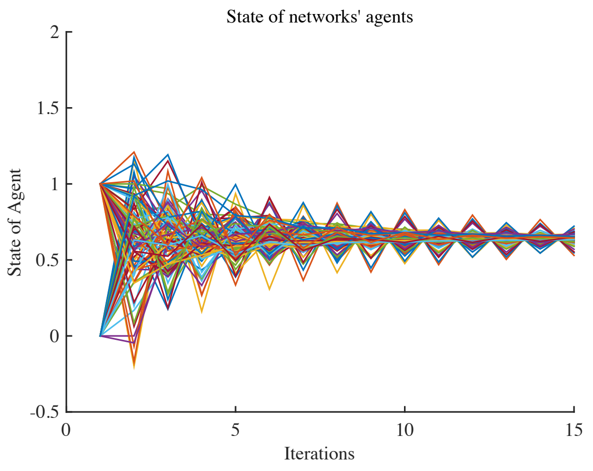


(a) Naive vs Cool Down optimization for a random vector.

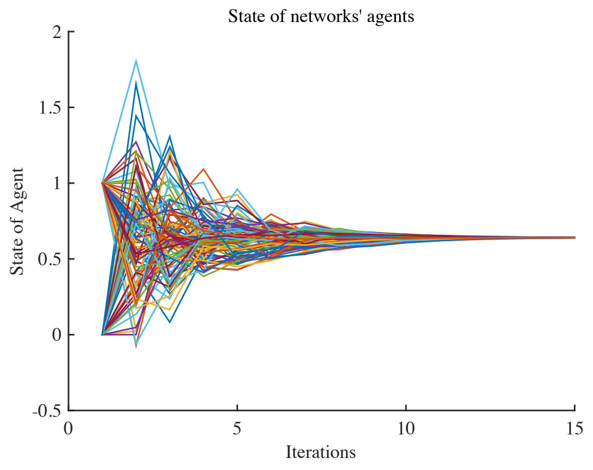


(b) Naive vs Cool Down optimization for a worst case vector.

Figure 1: State of agents as they communicate to reach consensus.

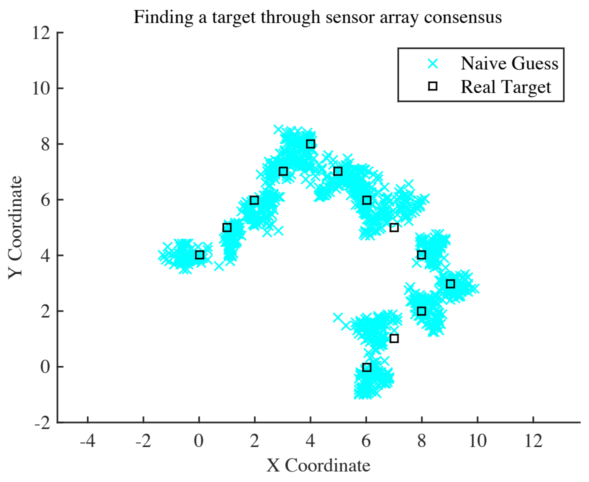


(a) Naive optimization.

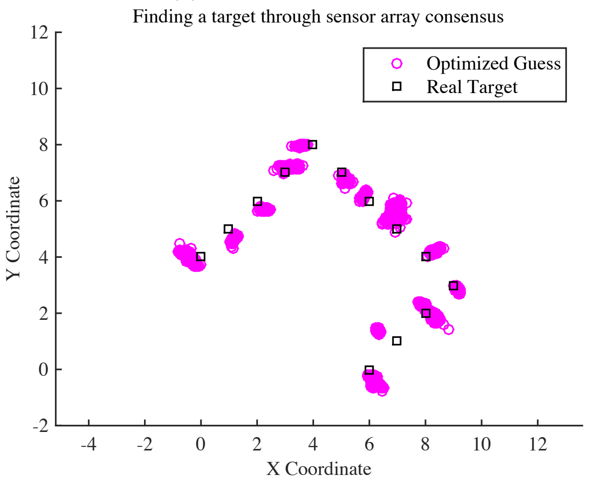


(b) Cool Down method.

Figure 2: State of agents as they communicate to reach consensus.



(a) Naive optimization.



(b) Cool Down method optimization.

Figure 3: Tracking a target through a sensor's network with different optimizations.

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 user the ability to pinpoint with some accuracy the path that was traveled by the object. Figures 3a and 3b compare the results obtained when a network of sensors tries to reach a consensus with a naively optimized network (3a) with the results from a network that was optimized through the best method found by the work done in this thesis (3b).

2. Part 1 of the thesis: designing sparse weight matrices for symmetric graphs

The standard Fast Distributed Linear Averaging (FDLA) problem designed in [4] does not take into account the cost associated with each communication channel of the network. This part of the work explored a formulation to reduce the number of connections used without severely impacting the performance.

2.1. Convex re-weighted l_1-norm Problem

Formulation. It is common to add a l_1 penalty based to the objective function of a given optimization problem, when we also desire sparse solutions. However, the re-weighted l_1-norm approach is more powerful than the l_1-norm and is based on the the logarithm function. For the problem at hand, it consists in adapting the FDLA formulation as follows:

$$\begin{aligned} W^{(t+1)} := \underset{W}{\text{argmin}} \quad & \left\| W - \frac{11^T}{n} \right\|_2 \\ & + \sum_{i,j} \frac{\beta}{|w_{ij}^{(t)}| + \varepsilon} |w_{ij}| \qquad (5) \\ \text{subject to} \quad & W = W^T \\ & W1 = 1 \\ & W \in \mathcal{S}, \end{aligned}$$

where $\beta > 0$ is a suitable constant, $\|\cdot\|_2$ denotes spectral norm, and $\mathcal{S}$ is the set of matrices with a zero in the (i, j)th position if no channel exists between agents i and j; the second term in the objective function arises from the linearization of $\log(|w_{ij}| + \varepsilon)$ at the the last iteration (in order to generate a convex problem). Optimization process (5) 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 FDLA problem. Being an iterative algorithm it needs a stopping criterion, which was defined as: $\left\|W^{(t+1)} - W^{(t)}\right\| / \left\|W^{(t)}\right\| \leq 0.05$.

Results. The first results obtained pointed to a problem in the formulation given by equation (5). 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 (which means consensus cannot be achieved). A new constraint was added to force the spectral norm to be within a given margin, e.g. 1%, of the FDLA performance:

$$\begin{aligned} W^{(t+1)} := \underset{W}{\text{argmin}} \quad & \left\| W - \frac{11^T}{n} \right\|_2 \\ & + \sum_{i,j} \frac{\beta}{|w_{ij}^{(t)}| + \varepsilon} |w_{ij}| \\ \text{subject to} \quad & W = W^T \\ & W1 = 1 \\ & \frac{\left\| W - \frac{11^T}{n} \right\|_2}{\left\| W^{(0)} - \frac{11^T}{n} \right\|_2} \leq 1.01 \\ & W \in \mathcal{S}. \end{aligned} \qquad (6)$$

This slight variation gives results that are capable of solving the consensus problem. Two formulations of problem (6) were considered: one where the diagonal entries of the matrix are considered ($i = j$ is allowed), and one where they are not ($i \neq j$ is imposed).

As an iterative problem, this formulation takes a lot longer to calculate than the previous formulations, but it also yields better results, as is is possible to see by Table 1.

Table 1: Cardinality of matrix W (number of nonzero entries) 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 4 shows the results obtained with several tolerances in the spectral norm constraint. The data labeled *diag* α% corresponds to the formulation that considers the diagonal entries, and the data labeled *nodiag* α% 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 blind to the cost, it will use every possible connection and self-weight optimally. This decrease happens in the *diag* α% 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 as well, worsening the performance. In the *nodiag* α% the diagonal entries have no cost, but still the simple FDLA formulation can achieve a small increase in performance.

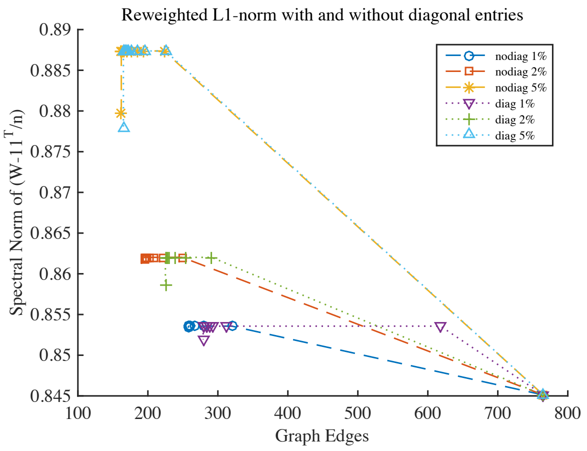


Figure 4: Spectral norm and number of communication channels active after the re-weighted l_1-norm problem.

2.2. 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. The two methods described are explained in more detail in the thesis: maximizing the Algebraic Connectivity 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 [3], where a bigger value means a better connectivity.

1st Problem statement. The work in [3] formulated a problem where the objective was to grow a graph to a desired number of connections. This was done under the premisse of maximizing the final connectivity, by choosing the best possible edges to do so. The method starts with a base Laplacian, $\mathbf{L}_{\text{base}}$ and a set of m edges from which we choose up to $k \leq m$ edges and add them to the graph. The question to answer is "What is the best set of edges to choose and add?".

To answer 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 $\mathbf{L}_{\text{base}} = AA^T = \sum_{l=1}^{m} a_l a_l^T$. This leads to the next formulation where, given a base Laplacian, a search is made to find the edges that maximize λ_2:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \lambda_2\left(\mathbf{L}_{\text{base}} + \sum_{l=1}^{k} a_l x a_l^T\right) \\ \text{subject to} \quad & \mathbf{1}^T x = k \\ & x_i \in \{0,1\}, \quad \text{for } i = 1,\dots,l. \end{aligned} \tag{7}$$

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. Problem (7) can be relaxed into one where x_i ranges between 0 and 1, which is a convex problem:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \lambda_2\left(\mathbf{L}_{\text{base}} + \sum_{l=1}^{k} a_l x a_l^T\right). \\ \text{subject to} \quad & \mathbf{1}^T x = k \\ & 0 \leq x_i \leq 1, \quad \text{for } i = 1,\dots,l. \end{aligned} \tag{8}$$

This problem can be formulated as an Semidefinite Programing (SDP) problem which is easy to solve numerically. The key idea is to exploit the Rayleigh-Ritz characterization of 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:

$$\begin{aligned} L(x) &= \begin{bmatrix} U & \frac{\mathbf{1}}{\sqrt{n}} \end{bmatrix} \begin{bmatrix} \lambda_n & & & \\ & \ddots & & \\ & & \lambda_2 & \\ & & & 0 \end{bmatrix} \begin{bmatrix} U^T \\ \frac{\mathbf{1}^T}{\sqrt{n}} \end{bmatrix} \\ &= U \begin{bmatrix} \lambda_n & & \\ & \ddots & \\ & & \lambda_2 \end{bmatrix} U^T + \frac{\mathbf{1}}{\sqrt{n}} \times 0 \times \frac{\mathbf{1}}{\sqrt{n}}. \end{aligned} \tag{9}$$

It is a well known fact in convex optimization that, when the objective is to maximize the smallest eigenvalue, the convex form of $\lambda_{\min}(A) \geq t$ 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,

$$\begin{aligned} & \begin{bmatrix} \lambda_n & & \\ & \ddots & \\ & & \lambda_2 \end{bmatrix} \succeq \begin{bmatrix} s & & \\ & \ddots & \\ & & s \end{bmatrix} \\ \Leftrightarrow\ & U \begin{bmatrix} \lambda_n & & \\ & \ddots & \\ & & \lambda_2 \end{bmatrix} U^T \succeq U \begin{bmatrix} s & & \\ & \ddots & \\ & & s \end{bmatrix} U^T \\ \Leftrightarrow\ & L(x) \succeq s\ UU^T. \end{aligned} \tag{10}$$

Equation $L(x) \succeq t\ UU^T$ is close to 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:

$$\begin{aligned} \begin{bmatrix} U & \frac{\mathbf{1}}{\sqrt{n}} \end{bmatrix} \begin{bmatrix} U^T \\ \frac{\mathbf{1}^T}{\sqrt{n}} \end{bmatrix} = I_n \Leftrightarrow UU^T + \frac{\mathbf{1}\mathbf{1}^T}{n} = I_n \\ \Leftrightarrow UU^T = I_n - \frac{\mathbf{1}\mathbf{1}^T}{n}. \end{aligned} \tag{11}$$

Finally, joining (10) and (11) the SDP formulation of problem (8) appears:

$$\begin{aligned} \underset{s,x}{\text{maximize}} \quad & s \\ \text{subject to} \quad & s\left(I - \mathbf{1}\mathbf{1}^T/n\right) \preceq L(x) \\ & \mathbf{1}^T x = k \\ & 0 \leq x_i \leq 1, \quad \text{for } i = 1,\dots,l. \end{aligned} \tag{12}$$

Though SDPs can easily be solved, the computational requirements of this approach motivated the authors in [3] 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 1. 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.

Algorithm 1 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
```

A modified version of this algorithm was also implemented based on the rollout idea [1], 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.

Dual Problem and dual Vairables. As explained in Boyd's "Convex Optimization" [2], each problem solved using Convex Optimization has associated with it a dual problem. 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 a minimum spanning tree (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 2.

Algorithm 2 Algorithm with Dual Variables to add edges.

```
Input a Laplacian;
Perform optimization on the $\rho(W - \mathbf{11^T}/n)$
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 $\rho(W - \mathbf{11^T}/n)$;
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.

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 5.

Figure 6 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 (12) can only produce sub-optimal solutions for the connectivity, which will reflect on the performance of the network. The heuristic shown in figure 7 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 8 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.

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.

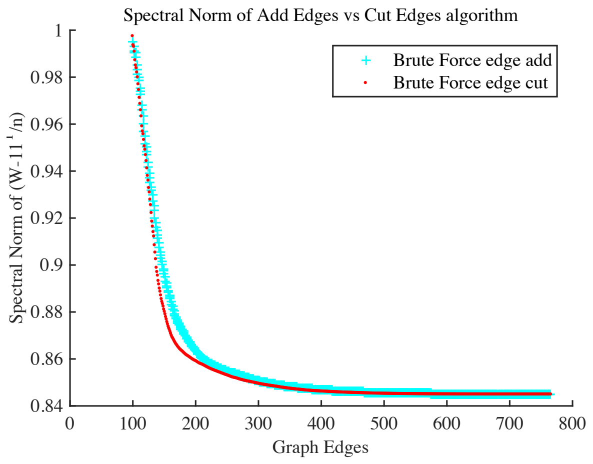


Figure 5: Greedy benchmark of the spectral norm compared with Brute Force growth.

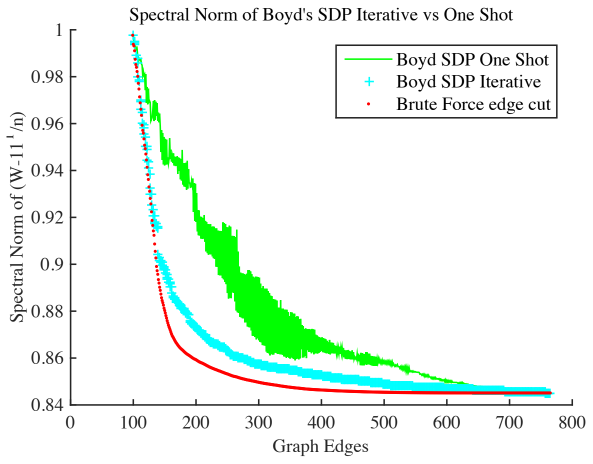


Figure 6: Greedy benchmark of the spectral norm compared with SDP growth

3. Part 2 of the thesis: designing weight matrices for asymmetric graphs

This section of the work explores the case where a communication network has channels that are unidirectional. Mathematically this translates into having an asymmetric matrix which, due to the non-convex nature of the spectral radius, is very difficult to optimize.

3.1. Minimizing the spectral norm

The spectral norm as a proxy for the spectral radius. Since the matrix W that represents

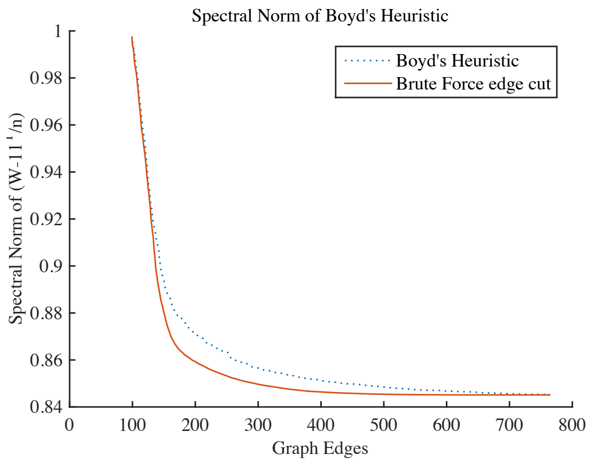


Figure 7: Greedy benchmark of the spectral norm compared with Heuristic growth.

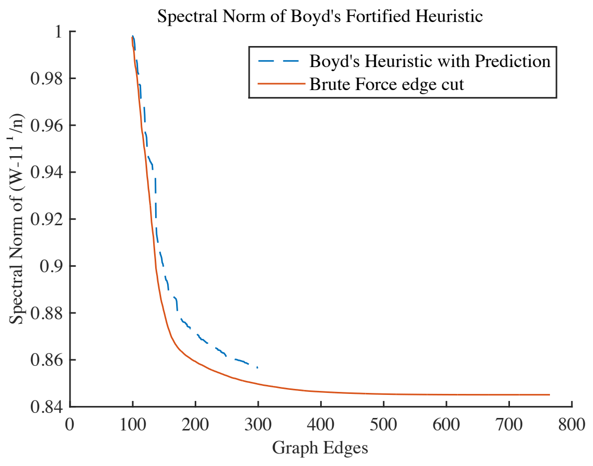


Figure 8: Greedy benchmark of the spectral norm compared with Fortified Heuristic growth.

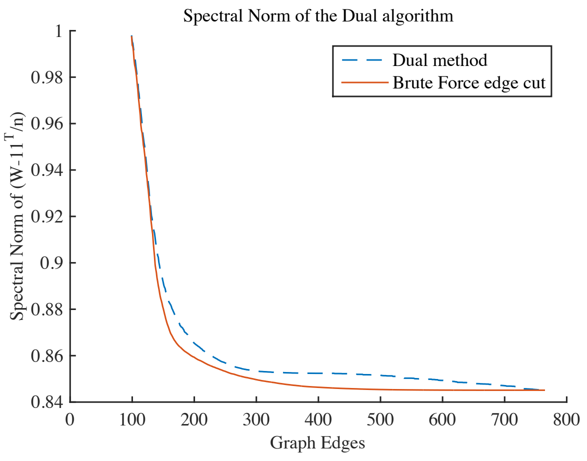


Figure 9: Greedy benchmark of the spectral norm compared with Dual growth.

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 (13) holds.

$$\rho(A) \leq \|A\|_2 \tag{13}$$

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:

$$\begin{array}{ll} \underset{W}{\text{minimize}} & \left\| W - \frac{1}{n}11^T \right\|_2 \\ \text{subject to} & 1^T W = 1^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{array} \tag{14}$$

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 2: 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 (14), 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, $\bar{W}$ = `randn(n,n)`—onto the feasible set, i.e., it solves the convex optimization problem:

$$\begin{array}{ll} \underset{W}{\text{minimize}} & ||W - \bar{W}||_2 \\ \text{subject to} & 1^T W = 1^T \\ & W1 = 1 \\ & W \in \mathcal{S}. \end{array} \tag{15}$$

These trials were preformed $2 \cdot 10^4$ times, and, for each matrix, its spectral radius was recorded. The results are displayed in Figure 10. 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 (14). This confirms that the approach in (14) does not lead to the fastest weigth matrix.

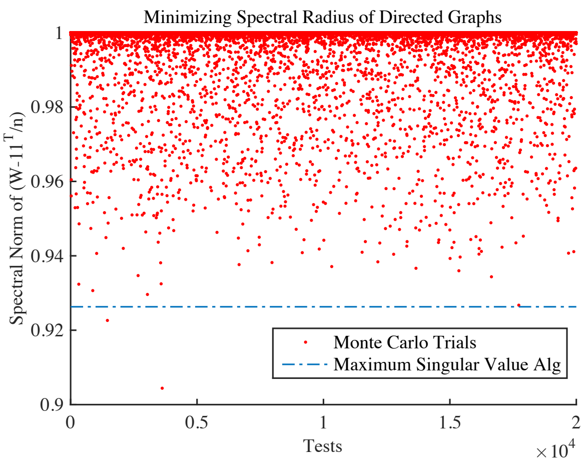


Figure 10: Monte Carlo trials to find W matrices that reach consensus.

3.2. Cool Down

In this section a new approach is developed based on

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

which states that the spectral radius can be verified by a matrix inequality after adding a new variable 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.

At each step, the faced feasibility problem is:

$$\begin{array}{ll} \text{Find} & P \text{ and } W \\ \text{subject to} & P \succ 0 \qquad W \in \mathcal{W} \\ & (W - J)^T P (W - J) \leq r^2 P. \end{array} \tag{16}$$

The problem is not convex 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,

$$\begin{aligned} & (W - \mathrm{J})^T P (W - \mathrm{J}) \preceq r^2 P \\ \Leftrightarrow\; & (W - \mathrm{J})^T Q^{-1} (W - \mathrm{J}) \preceq r^2 P \\ \Leftrightarrow\; & r^2 P - (W - \mathrm{J})^T Q^{-1} (W - \mathrm{J}) \succeq 0, \end{aligned} \tag{17}$$

resulting in

$$\begin{array}{ll} \text{Find} & P,\, W \text{ and } Q \\ \text{subject to} & P \succ 0 \qquad Q \succ 0 \\ & P = Q^{-1} \qquad W \in \mathcal{W} \\ & \begin{bmatrix} r^2 P & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q \end{bmatrix} \succeq 0. \end{array} \tag{18}$$

Since

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

equation (18) can be reformulated as

$$\begin{array}{ll} \text{Find} & P,\ W \text{ and } Q \\ \text{subject to} & P \succ 0 \qquad Q \succ 0 \\ & P \succeq Q^{-1} \qquad \mathrm{Tr}(PQ) = n \\ & W \in \mathcal{W} \\ & \begin{bmatrix} r^2 P & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q \end{bmatrix} \succeq 0. \end{array} \tag{19}$$

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

$$\begin{array}{ll} \underset{P,\ Q,\ W}{\text{minimize}} & \mathrm{Tr}(PQ) \\ \text{subject to} & P \succ 0 \qquad Q \succ 0 \\ & W \in \mathcal{W} \\ & \begin{bmatrix} P & I \\ I & Q \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} r^2 P & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q \end{bmatrix} \succeq 0. \end{array} \tag{20}$$

Thus, the goal becomes the minimization of $\mathrm{Tr}(PQ)$ over the remaining set of constraints; note that, for (P, Q, W) feasible in (20), 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 (20) is solved, the remaining test is to check if $\mathrm{Tr}(PQ) = n$ holds at the solution. If so, the problem (19) is feasible and the solution W from (20) 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:

$$\begin{array}{ll} \underset{W, \Delta_P, \Delta_Q}{\text{minimize}} & \mathrm{Tr}[(P + \Delta_P)\ Q + P\ (Q + \Delta_Q)] \\ \text{subject to} & \begin{bmatrix} r^2 (P + \Delta_P) & (W - \mathrm{J})^T \\ (W - \mathrm{J}) & Q + \Delta_Q \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P + \Delta_P & I_n \\ I_n & Q + \Delta_Q \end{bmatrix} \succeq 0 \\ & W \in \mathcal{W} \end{array} \tag{21}$$

and the final algorithm is

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:

$$\begin{array}{ll} \underset{k, P}{\text{minimize}} & k \\ \text{subject to} & I_n \leq P \leq k I_n \\ & (W_0 - J)^T P (W_0 - J) \leq (\eta^2 + \epsilon) P; \end{array} \tag{22}$$

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

4) Solve the following iterative algorithm:

$$\begin{array}{ll} \underset{W, \Delta_P, \Delta_Q}{\text{minimize}} & \mathrm{Tr}[(P + \Delta_P)\ Q + P\ (Q + \Delta_Q)] \\ \text{subject to} & \begin{bmatrix} \eta^2 (P + \Delta_P) & (W - J)^T \\ (W - J) & Q + \Delta_Q \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} P + \Delta_P & I_n \\ I_n & Q + \Delta_Q \end{bmatrix} \succeq 0 \\ & 1^T W = 1^T \\ & W 1 = 1 \\ & W \in \mathcal{S}; \end{array} \tag{23}$$

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

$$\frac{||\Delta_P||_F}{||P||_F} \leq 0.1 \quad \frac{||\Delta_Q||_F}{||Q||_F} \leq 0.1; \tag{24}$$

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

$$\eta = 0.95\eta; \tag{25}$$

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

Results. This algorithm achieved very good results, as shown in Figure 11. 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.

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. We developed efficient algorithms with complementary features. On one hand,

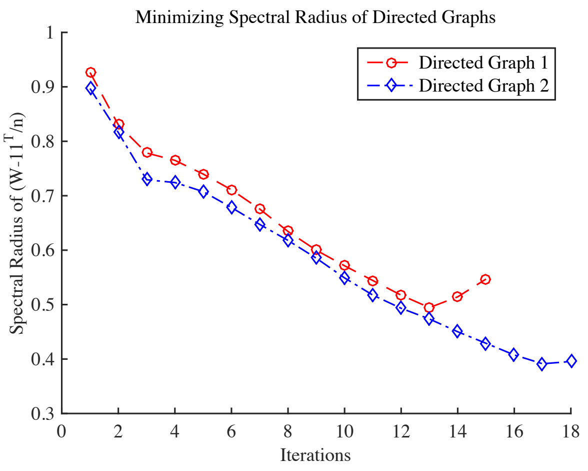


Figure 11: Cool Down algorithm results.

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 of the thesis, we 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.

Several methods were developed to tackle this problem, the most interesting of which being 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. This opens ground for future research, in the form of numerical refinement of the optimization methods developed in this thesis.

References

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

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

[3] A. Ghosh and S. Boyd. Growing well-connected graphs. In *Decision and Control, 2006 45th IEEE Conference on*, pages 6605–6611, Dec 2006.

[4] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems &amp; Control Letters*, 53(1):65–78, 2004.

[5] S. You. A fast linear consensus protocol on an asymmetric directed graph. In *2014 American Control Conference*, pages 3281–3286, June 2014.