Optimization, control, and game theoretical problems in consensus protocols

Mahmoud El Chamie

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS
ÉCOLE DOCTORALE DES SCIENCES ET TECHNOLOGIES DE
L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis
Mention : Informatique

Defended by

Mahmoud EL CHAMIE

Optimization, Control, and Game Theoretical Problems in Consensus Protocols

MAESTRO Team
(INRIA)
Advisors:

Konstantin Avrachenkov	INRIA Sophia Antipolis (France)
Giovanni Neglia	INRIA Sophia Antipolis (France)

Defended on 21 November, 2014

Jury:

Président:	Walid Dabbous	-	INRIA Sophia Antipolis (France)
Rapporteurs:	Pascal Bianchi	-	Télécom ParisTech (France)
	Mikael Johansson	-	KTH Royal Institute of Technology (Sweden)
Examinateurs:	Vivek Shripad Borkar	-	Indian Institute of Technology Bombay (India)
	Daniel Figueiredo	-	Federal University of Rio de Janeiro (Brazil)
	Sandro Zampieri	-	University of Padova (Italy)

Acknowledgements

This thesis has been accomplished with the help and support of a number of people that I would like to thank and acknowledge. I would like to express my deep gratitude to Dr. Giovanni Neglia, my PhD advisor, for guiding me through this period. His professional way in work by countless advice, feedback during discussions, and detailed review of our research reports guided my research. I would also like to thank my co-advisor Dr. Konstantin Avrachenkov for the discussions during our meetings. My deep appreciation and thanks go to professor Tamer Başar for hosting me 4 months in his team and treating me as one of his students at the University of Illinois at Urbana Champaign (UIUC).

I would also like to acknowledge the permanents in Maestro team (Philippe, Sara, Alain, and Eitan) for repeatedly providing their feedback on my work. Special thanks to Laurie (our team assistant) for helping us in filling the mission orders, going through all the administrative procedures, and helping me personally in french during my first year with the team. Thanks to Farah and Ali for reviewing the french translation of the introduction. I would like to acknowledge the committee members for spending their time in evaluating my work (special thanks to the reviewers of the manuscript, professor Pascal Bianchi and professor Mikael Johansson, for their detailed and rigorous review of the thesis). I would also like to thank Dr. Walid Dabbous for being the president of the jury and for supporting my Ubinet masters application (the first step that made this PhD possible).

Due to some people, INRIA has been a pleasant working environment and Juan les Pins has been a nice place to stay. I would like to thank Chadi for the coffee break chats we used to have. I would also like to extend my appreciation to my friends Alvinice and Khoa, I used to enjoy our weekly "Friday Ubinet lunch". Special thanks to my Juan les Pins neighbors: Ali, Rawad, and Salim for spending good moments together and finding good times away from work. I would also like to express my deep and sincere gratitude to Dana for her continuous and unlimited support all the time, during both health and sickness. To my family, you are my unwavering support and in spite of the distance, your prayers and encouragement guided me throughout all my life and made me what I am today and what I will be in the future.

Mahmoud El Chamie
10 November 2014
Sophia Antipolis, France

Optimization, Control, and Game Theoretical Problems in Consensus Protocols

Abstract: Consensus protocols have gained a lot of interest in the recent years. In this thesis, we study optimization, control, and game theoretical problems arising in consensus protocols.

First, we study optimization techniques for weight selection problems to increase the speed of convergence of discrete-time consensus protocols on networks. We propose to select the weights by applying an approximation algorithm: minimizing the Schatten p-norm of the weight matrix. We characterize the approximation error and we show that the proposed algorithm has the advantage that it can be either solved in a distributed way using a simple projected gradient method or solved by Newton's method and achieve faster convergence.

Then we propose a game theoretical framework for an adversary that can add noise to the weights used by averaging protocols to drive the system away from consensus. We give the optimal strategies for the game players (the adversary and the network designer) and we show that a saddle-point equilibrium exists in mixed strategies.

We also analyze the performance of distributed averaging algorithms where the information exchanged between neighboring agents is subject to deterministic uniform quantization (e.g., when real values sent by nodes to their neighbors are truncated). Using Lyapunov stability analysis, we characterize the convergence properties of the resulting nonlinear quantized system.

Consensus algorithms require that nodes exchange messages persistently to reach asymptotically consensus. The problem of termination of consensus protocols turns out to be challenging in the distributed setting. We propose a distributed algorithm for asymptotic termination of the consensus protocols. The algorithm reduces communication overhead while still guaranteeing convergence to consensus.

Finally, we propose a score metric that evaluates the quality of clusters such that the faster the random walk mixes in the cluster and the slower it escapes, the higher is the score. A local clustering algorithm based on this metric is proposed.

Keywords: Consensus Protocols; Distributed Averaging; Distributed Optimization; Multi-agent Systems; Game Theory; Adversarial Intervention; Quantization; Clustering.

Optimisation, Contrôle et Théorie des Jeux dans les Protocoles de Consensus

Résumé :

Les protocoles de consensus ont gagné beaucoup d'intérêt ces dernières années. Dans cette thèse, nous étudions les problèmes d'optimisation, de contrôle, et de théorie de jeu qui se posent dans ces protocoles.

Tout d'abord, nous étudions les techniques d'optimisation pour des problèmes de sélection de poids permettant ainsi d'augmenter la vitesse de convergence de protocoles de consensus dans les réseaux. Nous proposons de sélectionner les poids en appliquant un algorithme d'approximation: minimisation de la norme p de Schatten de la matrice de poids. Nous caractérisons l'erreur induite par cette approximation et nous montrons que l'algorithme proposé a l'avantage qu'il peut être soit résolu de façon distribuée en utilisant une méthode de gradient projeté simple ou résolu par la méthode de Newton et avec une convergence plus rapide.

Ensuite, nous proposons un cadre conceptuel d'analyse des jeux d'adversaire qui peut ajouter du bruit aux poids utilisés par l'algorithme de consensus de moyenne afin d'éloigner le système de consensus. Nous donnons les stratégies optimales pour les joueurs (l'adversaire et le concepteur du réseau) dans ce jeu et nous montrons qu'un point-selle (saddle-point equilibrium) existe en stratégies mixtes.

Nous analysons également la performance des algorithmes de consensus de moyenne où les informations échangées entre les agents voisins sont soumises à la quantification uniforme déterministe (les valeurs réelles envoyées par les nœuds de leurs voisins sont tronquées). En utilisant la notion de stabilité au sens de Lyapunov, nous caractérisons les propriétés de convergence du système quantifié non linéaire résultant.

Le problème de la terminaison des protocoles de consensus s'avère difficile dans le cadre distribué. Nous proposons un algorithme distribué pour la terminaison des protocoles de consensus. L'algorithme réduit la charge de communication tout en garantissant la convergence vers un consensus. Enfin, nous proposons une mesure de similarité qui évalue la qualité d'un regroupement (clustering) des nœuds dans un réseau. Un algorithme local de clustering basé sur cette métrique est donné.

Mots clés : Consensus de Moyenne; Calcul Distribué; Optimisation Distribuée; Systèmes Multi-Agents; Théorie des Jeux; Quantification; Regroupement.

Contents

Frequently Used Terms and Notation

Symbol	Description	Dimension
-	Vectors are usually denoted by small bold letters (e.g., $\mathbf{x}, \mathbf{w}, \dots$)	-
-	Matrices are usually denoted by capital letters (e.g., $X, W, \dots$)	-
G	network of nodes and links	-
V	set of nodes/vertices	$\|V\| = n$
E	set of links/edges	$\|E\| = m$
I_n	identity matrix	$n \times n$
$\mathbf{1}_n$	vector of all ones	$n \times 1$
D	degree diagonal matrix	$n \times n$
A	adjacency matrix of a graph	$n \times n$
Q	incidence matrix of a graph	$n \times m$
L	Laplacian matrix $L = D - A = QQ^T$	$n \times n$
$l \sim (i, j)$	link labeled l incident to nodes i and j	-
k	usually a discrete time index	integer
$\mathbf{x}(k)$	state vector of the system at iteration k	$n \times 1$
W	weight matrix (of the typical dynamics $\mathbf{x}(k+1) = W\mathbf{x}(k)$)	$n \times n$
$\mathbf{w}$	vector of weights on links	$m \times 1$
$\text{diag}(\mathbf{v})$	diagonal matrix having the elements of the $n \times 1$ vector $\mathbf{v}$	$n \times n$
$\mathcal{C}_G$	set of real matrices following G (having 0 at position (i, j) if $(i, j) \notin E$)	$n \times n$
λ_i	i-th largest eigenvalue ($\lambda_1 \geq \lambda_2 \geq \dots$)	scalar
Λ	eigenvalues diagonal matrix $\Lambda_{ii} = \lambda_i$	$n \times n$
σ_i	i-th largest singular value ($\sigma_1 \geq \sigma_2 \geq \dots$)	scalar
μ	second largest eigenvalue in magnitude of W	scalar
$\rho(X)$	spectral radius of matrix X	scalar
$\text{Tr}(X)$	trace of the matrix X	scalar
$\|\|X\|\|_{\sigma p}$	Schatten p-norm of a matrix X ($\|\|X\|\|_{\sigma p} = (\sum_i \sigma_i^p)^{1/p}$)	scalar
$O(.)$	Big-O notation (asymptotic notation)	-
$P_S(.)$	Projection on a set $S \subset \mathbb{R}^m$	$\mathbb{R}^m \to \mathbb{R}^m$

Chapter 1

Introduction

Contents

Who did not wonder how well interconnected we, human beings, are, not just with each other, but with the networked environment surrounding us. Most of the networks we face today are highly interconnected. The internet (connecting its users), the web (connecting its pages), communication networks, wireless sensor networks, smart grids, and more recently social networks are just few examples of interconnected environments. The interesting common feature to these networks is that they can be composed of many small subsystems taking local decisions (based only on neighboring interaction rules). These local decisions can have crucial impact on the entire network. For example, a virus spreading from an infected computer can lead to a serious damage in the network, and a video sharing by a well connected user in a social network can make the video go viral touching a large portion of the population.

In general, a network is formed of nodes (or agents) and communication links that allow these nodes to share information and resources. An agent in this thesis is a state machine (possibly an infinite state one) programmed to run algorithms according to well defined dynamics of interactions. These dynamics change the

states of agents (and thus the state of the system), and depending on local decisions these states can converge or not. The local decisions that cause the states of agents to converge to a common state are called *consensus protocols*. In this thesis we only consider discrete-time systems, but the rationale for the algorithms studied can be extended to continuous time systems as well.

Consensus protocols can be applied in various and broad network settings (as the ones mentioned earlier) where interactions between neighbors are possible. In fact, these protocols lie at the intersection of different research fields as systems theory, computational models, and graph theory. Systems theory is the trans-disciplinary study of the abstract organization of phenomena without being specific to an exact type of objects, to their exact properties, or to the qualitative description of their interaction rules in the underlying environment. This abstraction in consensus protocols is given by modeling the network by a graph of vertices (the agents) connected by edges (if they communicate), and then running consensus algorithms on the top of that.

As in any protocol, some parameters can be tuned in the consensus algorithm. Therefore optimizing the choice of these parameters leads to a better performance in terms of energy savings, speed of convergence, or robustness of the system to noise. In addition to optimization, controlling the states of the agents is very important. In some cases, bad choice of the parameters can cause the divergence of the states and destabilization of the system. In fact, designing local interaction rules for agents that provide some global guarantees is one of the main goals of the distributed optimization and control community when the agents are strategic participants. Game theory is a natural tool for analyzing these protocols and designing their interaction strategies for reaching a stabilizing state having some global optimization properties.

In this thesis, we investigate optimization problems concerned with discrete-time consensus protocols on networks, such as parameter tuning to increase speed of convergence, distributed implementation of global optimization problems, and minimization of the communication overhead (Chapters 2 and 5). We also propose a game theoretical framework to take into account an adversary in the network trying to disrupt the communication channel (Chapter 3). We design and analyze consensus algorithms in the presence of communication constraints as quantization (Chapter 4). We address the problem of detecting communities (clusters) in a network by proposing a novel scoring metric based on the speed of convergence of consensus protocols and the random walk spectral gap properties (Chapter 6).

The main motivation for this thesis is the following three applications where consensus protocols are a fundamental block in their design:

- Distributed optimization and control,
- Environmental monitoring in wireless sensor networks,
- Multi-agents coordination.

Thus the contribution of this thesis is to add knowledge to the research on consensus protocols in general and to these applications in particular.

Distributed Optimization and Control

There has been recently a significant amount of research on distributed optimization in networks. New faster techniques [WOJ13, GJS11] have been proposed for the traditional dual decomposition approach for separable problems that is well known in the network community since Kelly's seminal work on TCP [KMT98]. Other work in [NO09, JKJJ08] combines a consensus protocol, that is used to distribute the computations among the agents, and a subgradient method for the minimization of a local objective. A different approach relies on some intelligent random exploration of the possible solution space, e.g., using genetic algorithms [ANC+10] or the annealed Gibbs sampler [KBC+07]. In fact, distributed optimization by consensus protocols in the control community goes back to the 80's due to the work of D. P. Bertsekas and J. N. Tsitsiklis on decentralized decision making and parallel computing [BT89].

Consensus problems have also a close relationship with the PageRank algorithm used by Google search engine to rank the web pages of the search results [BP98]. Since the number of websites so far is more than 1 billion,[1] the PageRank requires the calculation of an eigenvector corresponding to the largest eigenvalue of extremely large but sparse matrix. Therefore, the use of global information is not feasible in this case, and distributed and parallel implementations are mandatory [LM06, ALNO07]. A possible way is by running consensus-like algorithms [IT10, ANP07]. The PageRank problem has recently been of the interest of the systems and control community [IT14].

In networks, algorithms for efficient routing and efficient use of resources are proposed to save energy and speed up the processing. For small networks, it is possible for a central unit to be aware of all the components of the network and decide how to optimally use a resource on a global view basis. As networks expand, the central unit needs to handle a larger amount of data, and centralized optimization may become unfeasible especially when the network is dynamic [BFH13]. In fact, the optimal configuration needs to be computed whenever a link fails or there is any change in the network. Moreover, nodes may have some processing capabilities that are not used in the centralized optimization. With these points in mind, it becomes more convenient to perform distributed optimization relying on local computation at each node and local information exchange between neighbors [Joh08]. Such distributed approach is intrinsically able to adapt to local network changes.

Environmental Monitoring in Wireless Sensor Networks

Emerging technologies as robotics, multi-vehicle cooperation control, and environmental monitoring have a driving need for wireless sensor networks. In these networks, a group of sensors communicates in an ad-hoc manner to accomplish the

[1] www.internetlivestats.com

tasks they are deployed to do.

Environmental monitoring requires that sensors measure temperature, pressure, pollution, etc. in their area of deployment. These measurements can be noisy and if, for example, the noise is additive, zero mean, and Gaussian, then each temperature sensor can have a different noisy measurement of the nominal temperature. It is well known that a good filter of the Gaussian noise (achieving the maximum likelihood) is the mean filter. Therefore, averaging the values of the initial measurements can give a more accurate estimation, this is known as *sensor fusion*. Sensor fusion can be obtained by decentralized communication between sensors by consensus protocols. In fact, sensor fusion is the motivation provided by Boyd *et al.* for their well known paper on gossiping consensus protocols [BGPS06].

Some computational models are also motivated by wireless sensor consensus applications. The proposed model, the *population protocols*, was first introduced in [AAD+04] as a model for distributed (computational capable) agents interacting locally to infer some global information about the group. This model is motivated by sensors attached to birds in a flock with the goal to check some global properties relying only on local interactions, like determining whether more than 5% of the population has elevated temperature.

Multi-agents Coordination

Consensus protocols find their way also in multi-agents coordination problems [OSFM07]. The agents in such networks also use the wireless sensor technology to communicate. A group of robots moving in parallel for example should agree on the direction of motion and the speed to avoid collision. In formation control problems, with a leaderless approach, robots only communicate on a neighbor to neighbor basis to collectively accomplish a global task [JLM03, BA98] (like obstacle avoidance or trajectory following while maintaining connectivity [JE07]). The main difficulty for the consensus protocols in this category of problems does not originate from the large number of robots, but rather from the switching topology and connectivity issues.

1.1 Background

1.1.1 The Network Model

Consider a network of n nodes that can exchange messages between each other through communication links. The network of nodes can be modeled as a graph $G = (V, E)$ where V is the set of vertices, labeled from 1 to n, and E is the set of edges, labeled from 1 to m. $(i, j) \in E$ if nodes i and j are connected and can communicate (they are neighbors). If link (i, j) has label l, we write $l \sim (i, j)$.[2]

[2]Most of the work in this thesis deals with static graphs, however some of the results can be naturally extended to include a dynamic graph topology.

Unless otherwise specified, graphs are considered to be *connected* and *undirected*.[3] Denote by d_i the degree of node i in the graph G.

Any given graph $G = (V, E)$ can be represented and fully characterized using one of the following matrices: the adjacency matrix A, the incidence matrix Q, or the Laplacian matrix L. The adjacency matrix A is the symmetric n by n square matrix whose elements are given as follows,

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{else.} \end{cases} \tag{1.1}$$

The incidence matrix Q is the n by m matrix where each column $k \sim (i,j)$ corresponds to a link and each column has only two nonzero elements,

$$\begin{cases} Q_{ik} = \pm 1 \\ Q_{jk} = -Q_{ik}, \end{cases} \tag{1.2}$$

The value of Q_{ik} can be either $+1$ or -1 because the graph is undirected. Finally, the Laplacian matrix is the symmetric n by n square matrix such that,

$$L_{ij} = \begin{cases} -1 & \text{if } (i,j) \in E, \\ d_i & \text{if } i = j, \\ 0 & \text{else.} \end{cases} \tag{1.3}$$

These matrices are related by the following formula,

$$L = D - A = QQ^T,$$

where D is the degree diagonal matrix ($D_{ii} = d_i$ for all $i \in V$). From the given definition we can deduce some properties of the Laplacian, since $L = QQ^T$, then it is a positive semi-definite matrix having nonnegative eigenvalues. Given that $L\mathbf{1} = \mathbf{0}$, where $\mathbf{1}$ is a vector of all ones, and $\mathbf{0}$ is the vector of all zeros, then 0 is an eigenvalue and $\frac{1}{\sqrt{n}}\mathbf{1}$ is the corresponding right unit eigenvector. Since the network is connected, it is well known that the second smallest eigenvalue of the Laplacian is strictly positive (and is called the algebraic connectivity [Fie73]). These matrices will appear often in this thesis.

Since most of the results in this thesis are theoretical, simulations and performance evaluation are done to support the theoretical findings. We mainly relied on connected random graphs,[4] so we give here an overview of these random networks:

- Random Geometric Graphs (RGGs) [Pen03] where n nodes are placed uniformly at random on a convex unit area (we considered a unit square area), and any two nodes are connected by an edge if the distance between them is

[3] Since the graph is undirected graph, then $(i,j) = (j,i)$ are eventually the same link.

[4] In some cases we also did simulations on real networks as Enron company internal email exchange network [SA04] and the dolphin social network [LSB+03], or static networks as grids or rings.

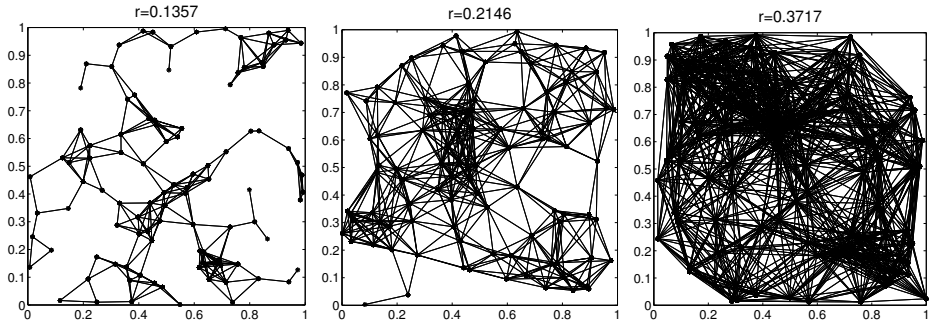

Figure 1.1: RGG with $n = 100$ nodes and different values of the connectivity radius.

less than the radius $r_n = \sqrt{c \times \frac{\log(n)}{n}}$, where c is a constant, see Fig. 1.1. The connectivity of RGG graphs is usually studied as a function of the scalar c [GK98]. RGGs are well suited to model wireless sensor networks where the nodes have been deployed randomly on a field and the transmission range of each sensor is r_n. When the transmission range r_n is small, the network presents clusters of nodes.

- Erdős-Rényi (ER) graphs, these graphs have a parameter P for the probability that a link to exist between any pair of vertices. They are constructed as follows: starting from an n-nodes-fully-connected graph, every link can be removed from the graph with probability $1-P$ and is left there with probability P.

1.1.2 Average Consensus

The graph $G = (V, E)$ is an abstraction of the network topology connecting communicating agents. Each agent (or node) in V can only communicate with its "neighbors". Neighbor relations are described as follows: agent j is a *neighbor* of agent i if (i, j) is an edge of G, i.e., $(i, j) \in E$. We denote by N_i the neighborhood set of node i. Every node i in the network has control over a real-valued scalar quantity x_i called an *agreement variable* whose value can be updated by the agent from time to time. Initially each agent i has a real scalar value $x_i(0) \in \mathbb{R}$. Let

$$x_{ave} = \frac{1}{n} \sum_{i \in V} x_i(0),$$

be the average of initial values of all agreement variables in the network. The purpose of the average consensus (distributed averaging) problem is to devise an algorithm for each agent which enables all n agents to asymptotically determine in a decentralized manner, the average of the initial values of their scalar variables, i.e.,

$$\lim_{k \to \infty} x_i(k) = x_{ave}. \tag{1.4}$$

A well studied approach to the problem is for each agent to use a linear iterative update rule of the form

$$x_i(k+1) = w_{ii}(k)x_i(k) + \sum_{j \in N_i} w_{ij}(k)x_j(k), \quad \forall i \in V, \tag{1.5}$$

where k is a discrete time index, and $w_{ij}(k)$ are real-valued weights to be designed (in general they are time varying specially for dynamic networks). Equation (1.5) can be written in a matrix form as

$$\mathbf{x}(k+1) = W(k)\mathbf{x}(k), \tag{1.6}$$

where $\mathbf{x}(k)$ is the state vector of agreement values whose i-th element is $x_i(k)$, and $W(k)$ is the weight matrix whose ij-th entry equals $w_{ij}(k)$. Equation (1.5) is the general state dynamical equation of consensus protocols that will appear often along this thesis. Therefore, it is important to understand the conditions on the matrix $W(k)$ that cause the states to converge (and more specifically to converge to the average consensus given by equation (1.4)). Let us first introduce some notation. We denote by λ_i the i-th eigenvalue of a matrix. For real and symmetric matrices, all eigenvalues are real and hence we can order them ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$), so λ_i is the i-th *largest* eigenvalue of the matrix. Denote by $\rho(.)$ the spectral radius of a matrix, i.e., $\rho = \max_i |\lambda_i|$, and by μ the largest eigenvalue in module non considering λ_1, i.e., $\mu = \max\{\lambda_2, -\lambda_n\}$ when the eigenvalues are all real. σ_i is the i-th largest singular value of a matrix, i.e., $\sigma_i(X) = \sqrt{\lambda_i(X^T X)}$. Notice that $\sigma_1(X) = ||X||_2$ where $||.||_2$ is the matrix L_2-induced norm.[5]

1.1.3 Convergence Conditions

Assume that $W(0), W(1), \ldots$ are independent and identically distributed random matrices, then sufficient conditions for almost sure convergence to consensus starting from any initial condition are [Bén09]:

$$\mathbf{1}^T W(k) = \mathbf{1}^T, \quad \text{for all } k, \tag{1.7}$$

$$W(k)\mathbf{1} = \mathbf{1}, \quad \text{for all } k, \tag{1.8}$$

$$\lambda_2(\mathbb{E}[W(k)^T W(k)]) < 1. \tag{1.9}$$

As we will be working on static graphs, then it is important to study the conditions for convergence when the same matrix is applied at all iterations, i.e., $W(k) = W$ for all k. In this case, [XB04] provides the following set of necessary and sufficient conditions that guarantee convergence to consensus starting from any initial condition:

$$\mathbf{1}^T W = \mathbf{1}^T, \tag{1.10}$$

$$W\mathbf{1} = \mathbf{1}, \tag{1.11}$$

$$\rho(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T) < 1. \tag{1.12}$$

[5] A matrix L_p-induced norm is defined as follows: $||X||_p = \max\{||X\mathbf{y}||_p : \mathbf{y} \in \mathbb{K}^n \text{ with } ||\mathbf{y}||_p = 1\}$, where $\mathbb{K}$ is a field of real numbers and $||\mathbf{y}||_p = \left(\sum_i |y_i|^p\right)^{1/p}$ is the usual L_p-norm of a vector.

Note that with these conditions, $\rho(W-\frac{1}{n}\mathbf{1}\mathbf{1}^T) = \mu(W)$ and $\sigma_1(W-\frac{1}{n}\mathbf{1}\mathbf{1}^T) = \sigma_2(W)$. It is well known that for any matrix X, the following holds [Ber05, p. 351]:

$$\rho(X) \leq \sigma_1(X),$$

where the equality holds if X is a symmetric matrix. Therefore, $\rho(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T) \leq \sigma_2(W)$ and thus if (1.9) is satisfied, then so is (1.12), but the inverse is not always true unless W is symmetric ($W = W^T$). We also observe that the weights are not required to be non-negative. Since we will mainly focus on problems where W is symmetric, then the first two conditions are equivalent to each other and equivalent to the possibility to write the weight matrix as follows:

$$W = I - Q \times \text{diag}(\mathbf{w}) \times Q^T, \tag{1.13}$$

where I is the identity matrix and $\mathbf{w} \in \mathbb{R}^m$ is the vector of all the weights on links w_l, $l = 1, \ldots, m$. Equation (1.13) gives an important representation of the weights, giving a relation between, on one hand, the weights on links in $\mathbb{R}^m$ and, on the other hand, the weight matrix in $\mathbb{R}^{n,n}$. To show the importance of this equation, suppose we have an optimization problem where the elements of the weight matrix W are the variables of this problem, then the number of variables is n^2. By applying equation (1.13), then there will be m variables which guarantees complexity savings specially on sparse graphs where $m = O(n)$.

1.1.4 Fastest Consensus

The system equation (1.6) for fixed weight matrix has a solution given as follows:

$$\mathbf{x}(k) = W^k\mathbf{x}(0). \tag{1.14}$$

The speed of convergence of the system given in (1.14) is governed by how fast W^k converges. For a real symmetric weight matrix, W has real eigenvalues and it is diagonalizable. We can write W^k using the orthonormal decomposition as follows [Mey00, p. 517]:

$$W^k = \sum_i \lambda_i^k G_i, \tag{1.15}$$

where $G_i = \mathbf{v}_i\mathbf{v}_i^T$ with $\mathbf{v}_i$ being the eigenvector corresponding to the eigenvalue λ_i. We note that the matrices G_is have the following properties: G_i is the projector onto the null-space of $W - \lambda_i I$ along the range of $W - \lambda_i I$, $\sum_i G_i = I$ and $G_iG_j = 0^{n\times n} \quad \forall i \neq j$. Conditions (1.10)-(1.12) imply that 1 is the largest eigenvalue of W in module and is simple. Then $\lambda_1 = 1$, $G_1 = 1/n\mathbf{1}\mathbf{1}^T$ and $|\lambda_i| < 1$ for $i > 1$. From the above representation of W^k, we can deduce two important facts:

1. First we can check that W^k actually converges, in fact we have $\lim_{k\to\infty}\mathbf{x}(k) = \lim_{k\to\infty} W^k\mathbf{x}(0) = \frac{1}{n}\mathbf{1}\mathbf{1}^T x(0) = x_{ave}\mathbf{1}$ as expected.

2. Second, the speed of convergence of W^k is governed by the second largest eigenvalue in module, i.e., on $\mu = \max\{\lambda_2, -\lambda_n\} = \rho\,(W - G_1)$. For obtaining the fastest convergence, nodes have to select weights that minimize μ, or equivalently maximize the spectral gap[6] of W.

Then the problem of finding the weight matrix that guarantees the fastest convergence can be formalized as follows:

$$\begin{aligned} \underset{W}{\text{argmin}} \quad & \mu(W) \\ \text{subject to} \quad & W = W^T, \\ & W\mathbf{1} = \mathbf{1}, \\ & W \in \mathcal{C}_G, \end{aligned} \tag{1.16}$$

where the last constraint on the matrix W derives from the assumption that nodes can only communicate with their neighbors and then necessarily $w_{ij} = 0$ if $(i, j) \notin E$. Problem (1.16) is called in [XB04] the "symmetric FDLA problem."

The above minimization problem is a convex one and the function $\mu(W)$ is non-smooth convex function. It is convex since when W is a symmetric matrix, we have $\mu(W) = \rho(W - G_1) = ||W - G_1||_2$ which is a composition between an affine function and the matrix L-2 norm, and all matrix norms are convex functions. The function $\mu(W) = \rho(W - G_1)$ is non-smooth since the spectral radius of a matrix is not differentiable at points where the eigenvalues coalesce [FN95]. The process of minimization itself in (1.16) tends to make them coalesce at the solution.

Moreover, the weight matrix solution of the optimization problem is not unique. For example it can be checked that for the network in Fig. 1.2, there are infinite weight values that can be assigned to the link $(2, 3)$ and solve the optimization problem (1.16), including $w_{23} = 0$. Additionally, this shows that adding an extra link in a graph (e.g., link $(2, 3)$ in the Fig. 1.2), does not necessarily reduce the second largest eigenvalue of the optimal weight matrix.

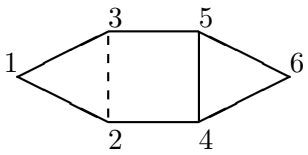

Figure 1.2: Network of 6 nodes.

[6] The spectral gap is the difference between the largest eigenvalue in module and the second largest one in module. In this case it is equal to $1 - \mu$.

1.2 Contributions

On the basis of the model equation (1.5), we study in this thesis optimization, control, and game theoretical problems that may arise. In particular, we raise the following questions:

- Given that the optimal weights to speed up convergence are known (by solving (1.16) globally), can the nodes in the network eventually infer (or learn) these weights in a distributed way (without the need of the global knowledge of the network)?
- Given that the network can be susceptible to attacks from an adversary willing to drive the system away from consensus, what strategies should be used by both, the adversary and the network designer, to achieve their opposite objectives?
- Suppose that communication channels between nodes in the network are subject to bandwidth constraints, and nodes can only receive/send truncated values of their neighbors' variables. How does this quantization affect the convergence of the resulting *nonlinear* system?
- Given that the state variables are converging asymptotically, can the nodes know when their state variables are close enough to the asymptotic value, and thus decide to stop executing the algorithm only on the bases on some local knowledge?
- Given that nodes forming well connected groups in the graph have similar convergence dynamics of their state variables, can we use this observation to identify these clusters of nodes?

Each chapter in the thesis deals with one of the above questions. Rather than mentioning here all the previous literature on consensus protocols as a general topic, we preferred to give the related works in every chapter specific to the problem studied. Below we list the contribution detailed in the following chapters.

1.2.1 Weight Optimization in Consensus Protocols

As we mentioned in Section 1.1.4, the convergence rate of the linear consensus algorithm is determined by the second largest eigenvalue in module of the weight matrix W. Optimal weights can be calculated by optimizing spectral properties of the weight matrix as in problem (1.16), which can be solved by semi-definite programming (SDP) as shown by Xiao *et al.* in [XB04]. The SDP cannot be implemented by the nodes in the network unless they have the full knowledge of the network. For this reason we propose to select the weights through an approximation algorithm which minimizes the Schatten p-norm of the weight matrix (essentially minimizing the trace of W^p). We characterize the approximation error and we show that the approximated solution has the advantage that it can be calculated in a

distributed way using a simple projected gradient method. We also provide a faster Newton's method to determine it.

The publications directly related to this contribution are the following:

- [ECNA15] M. El Chamie, G. Neglia, and K. Avrachenkov, "Distributed Weight Selection in Consensus Protocols by Schatten Norm Minimization", To appear in IEEE Transactions on Automatic Control as Technical Note, Volume 60, No. 4, April 2015.

- [ECN14] M. El Chamie and G. Neglia, "Newton's Method for Constrained Norm Minimization and Its Application to Weighted Graph Problems", In proceedings of the American Control Conference ACC 2014 (Portland, OR, United States, June 4-6), pp. 6, June 2014.

Other publications also related to this topic are the following:

- [SECN13] L. Severini, M. El Chamie, and G. Neglia, "Topology versus Link Strength for Information Dissemination in Networks", In proceedings of ALGOTEL 2013 (Pornic, Loire-Atlantique, France, May 28-31), pp. 4, May 2013.

- [AECN11] K. Avrachenkov, M. El Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks", In proceedings of IEEE International Conference on Distributed Computing in Sensor Sytems and Workshops DCOSS 2011 (Barcelona, Spain June 27-29), pp. 6, June 2011.

1.2.2 Adversarial Intervention

In this work, we propose a game theoretical framework for an adversary that can add noise to the weights used by averaging protocols to drive the system away from consensus. We give the equilibrium strategies for the players (the adversary and the network designer) in this game and we show that a saddle-point equilibrium (SPE) does not exist in pure strategies but it does in mixed strategies. We also study dynamic optimal weight selection optimal control for consensus protocols. For the multi-stage case, the solution exists but can rarely be expressed in closed-form equations. In view of this, we apply optimization techniques to obtain a locally (and possibly globally) optimizing feasible control path. For the one-stage case, we obtain a closed-form solution for the optimal control and provide sufficient conditions for the existence of a control that makes the system reach consensus in only one iteration.

The publication related to this contribution is the following:

- [ECB14] M. El Chamie and T. Başar, "Optimal Strategies for Dynamic Weight Selection in Consensus Protocols in the Presence of an Adversary", Accepted to the 53rd IEEE Conference on Decision and Control CDC 2014 (Los Angeles, California, Dec. 15-17), pp. 6, Dec. 2014.

1.2.3 Quantized Communication

We analyze the performance of distributed averaging algorithms where the information exchanged between neighboring agents is subject to deterministic uniform quantization (i.e., real values sent by nodes to their neighbors are truncated). With such quantization, convergence to the precise average cannot be achieved in general, but the convergence would be to some value close to it, called quantized consensus. Using Lyapunov stability analysis, we characterize the convergence properties of the resulting nonlinear quantized system. We show that in finite time and depending on initial conditions, the algorithm will either cause all n agents to reach a quantized consensus where the consensus value is the largest quantized value not greater than the average of their initial values, or will lead all n variables to cycle in a small neighborhood around the average. In the latter case, we identify tight bounds for the size of the neighborhood and we further show that the error can be made arbitrarily small by adjusting the algorithm's parameters in a distributed manner.

The publication related to this contribution is the following:

- [ECLB14] M. El Chamie, J. Liu, and T. Başar, "Design and Analysis of Distributed Averaging with Quantized Communication", Accepted to the 53rd IEEE Conference on Decision and Control CDC 2014 (Los Angeles, California, Dec. 15-17), pp. 6, Dec. 2014.

1.2.4 Reducing Communication Overhead

Consensus algorithms require that nodes exchange messages persistently to reach asymptotically consensus. The problem of termination of consensus protocols turns out to be challenging in the distributed setting. We propose a totally distributed algorithm for average consensus where nodes send more messages when they have large differences in their estimates, and reduce their message sending rate when the consensus is almost reached. The convergence of the system is guaranteed to be within a predefined margin η. Tuning the parameter η provides a trade-off between consensus precision and communication overhead of the protocol. The proposed algorithm is robust against nodes changing their initial values and can also be applied in dynamic networks with faulty links.

The publication related to this contribution is the following:

- [ECNA13] M. El Chamie, G. Neglia, and K. Avrachenkov, "Reducing Communication Overhead for Average Consensus", In proceedings of IFIP Networking 2013 (Brooklyn, NY, USA, May 22-24), May 2013.

1.2.5 Detecting Communities

Finally, we study the problem of finding well connected clusters (of nodes) in a network. It is well known that the mixing time of a random walk on a network is related to the speed of convergence of consensus protocols. We propose a score metric that evaluates the quality of clusters such that the faster the random walk

mixes in the cluster and the slower it escapes, the higher is the score. A local clustering algorithm based on this metric is given.

The publication related to this contribution is the following:

- [AECN14] K. Avrachenkov, M. El Chamie, and G. Neglia, "Graph Clustering Based on Mixing Time of Random Walks", In proceedings of the IEEE International Conference on Communications ICC 2014 (Sydney, Australia, June 10-14), pp. 6, June 2014.

1.2.6 Open Research Direction

We further introduce in the Appendix, as an open future research direction for consensus protocols, a novel model for averaging on networks with dynamic nodes. In dynamic networks, the network topology in the network changes with time. This can be due to mobility, link failure, or node failure. Most of the work on consensus in dynamic network settings consider a fixed number of nodes trying to reach agreement in the presence of mobility or non-robust links (so only the links are dynamic). However, little study has been made on networks with dynamic number of nodes. In this chapter, we study this problem on simple graph topology networks (like complete graphs and trees) and we leave the full model study as a future open research direction.

This work is due to the following presentation:

- M. El Chamie, G. Neglia, and K. Avrachenkov, "Averaging on Dynamic Networks", 10ème Atelier en Evaluation de Performances (Inria, Sophia Antipolis, 11-13 juin), June 2014. (abstract)

Chapter 2

Weight Optimization in Consensus Protocols

Contents

The speed of convergence of average consensus protocols depends on the weights selected on links (to neighbors). We address in this chapter how to select the weights in a given network in order to have a fast convergence speed for these protocols. We approximate the problem of optimal weight selection by the minimization of the Schatten p-norm of a matrix with some constraints related to the connectivity of the underlying network. We first provide a methodology for solving the Schatten p-norm optimization using the Newton's method. We then provide a totally distributed gradient method to solve the Schatten p-norm optimization problem. By tuning the parameter p in our proposed distributed minimization, we can simply trade-off the quality of the solution (i.e., the speed of convergence) for communication/computation requirements (in terms of number of messages exchanged and volume of data processed). The weight optimization iterative procedure can also run in parallel with the consensus protocol and form a joint consensus–optimization procedure.

2.1 Optimization Problem

We start by introducing formally the problem studied in this chapter. As mentioned in the introduction, the network of nodes can be modeled as a graph $G = (V, E)$ where V is the set of vertices, labeled from 1 to n, and E is the set of edges, then $(i, j) \in E$ if nodes i and j are connected and can communicate (they are neighbors) and $|E| = m$. We label the edges from 1 to m. If link (i, j) has label l, we write $l \sim (i, j)$. N_i is the neighborhood set of node i. All graphs in this chapter are considered to be *connected* and *undirected*. Let $x_i(0) \in \mathbb{R}$ be the initial value of the local variable at node i. We are interested in computing the average

$$x_{ave} = (1/n) \sum_{i=1}^{n} x_i(0),$$

in a decentralized manner with nodes only communicating with their neighbors. The network is supposed to operate synchronously: when a global clock ticks, all nodes in the system perform the iteration of the averaging protocol. At iteration $k + 1$, node i updates its state value x_i as follows:

$$x_i(k + 1) = w_{ii} x_i(k) + \sum_{j \in N_i} w_{ij} x_j(k). \tag{2.1}$$

As it is commonly assumed, in this chapter we consider that two neighbors select the same weight for each other, i.e., $w_{ij} = w_{ji}$. The matrix form equation is:

$$\mathbf{x}(k + 1) = W\mathbf{x}(k), \tag{2.2}$$

where $\mathbf{x}(k)$ is the state vector of the system and W is the weight matrix. The main problem we are considering in this chapter is how a node i can choose the weights w_{ij}

for its neighbors so that the state vector $\mathbf{x}$ of the system converges fast to consensus. As we have seen in the introduction, the necessary and sufficient conditions for the convergence of the system to average consensus starting from any initial value are the following:

$$\mathbf{1}^T W = \mathbf{1}^T, \tag{2.3}$$
$$W\mathbf{1} = \mathbf{1}, \tag{2.4}$$
$$\mu(W) < 1, \tag{2.5}$$

where $\mu(W) = \rho(W - G_1)$ is the second largest eigenvalue of W in module and $G_1 = \frac{1}{n}\mathbf{1}\mathbf{1}^T$. For symmetric weight matrices, the problem of finding the weight matrix that guarantees the fastest convergence, also given in the Introduction, can be formalized as follows:

$$\begin{aligned} \underset{W}{\text{argmin}} \quad & \mu(W) \\ \text{subject to} \quad & W = W^T, \\ & W\mathbf{1} = \mathbf{1}, \\ & W \in \mathcal{C}_G, \end{aligned} \tag{2.6}$$

where the last constraint on the matrix W derives from the assumption that nodes can only communicate with their neighbors and then necessarily $w_{ij} = 0$ if $(i, j) \notin E$. The constraint $W = W^T$ in the optimization requires any two neighbors i and j to choose the same weight on their common link $l \sim (i, j)$ i.e., $w_{ij} = w_{ji} = w_l$. The condition $W\mathbf{1} = \mathbf{1}$ means that at every node i the sum of all weights on its incident links plus its self-weight w_{ii} must be equal to one. This condition is satisfied if nodes choose first weights on links, and then adapt consequently their self-weights w_{ii}. Thus all three constraints in (2.6) lead to the possibility to write W as follows:

$$W = I - Q \times \text{diag}(\mathbf{w}) \times Q^T, \tag{2.7}$$

where $\mathbf{w} \in \mathbb{R}^m$ is the vector of all the weight links w_l, $l = 1, \ldots, m$, and Q is the incidence matrix of the graph (given in the introduction chapter as one of the three matrices that characterize the nodes and links in a graph). Problem (2.6) is called in [XB04] the "symmetric FDLA problem."

2.2 Related Work

Xiao and Boyd in [XB04] have shown that the symmetric FDLA problem (2.6) can be formulated as a Semi-Definite Program (SDP) that can be solved by a centralized unit using interior point methods. The semi-definite program is the following:

$$\begin{aligned} \underset{\mathbf{w},\, s}{\text{minimize}} \quad & s \\ \text{subject to} \quad & -sI \preceq I - Q \times \text{diag}(\mathbf{w}) \times Q^T - G_1 \\ & I - Q \times \text{diag}(\mathbf{w}) \times Q^T - G_1 \preceq sI, \end{aligned} \tag{2.8}$$

where s is an auxiliary real optimization variable, $A \preceq B$ if and only if $B - A$ is positive semi-definite, and $G_1 = \frac{1}{n}\mathbf{1}\mathbf{1}^T$. The output of this program is the optimal weight vector $\mathbf{w} \in \mathbb{R}^m$ such that $w_l, l = 1, \ldots, m$ is the weight selected for link l. The weight matrix can be then deduced from $\mathbf{w}$ using Eq. (2.7).

The limit of such centralized approach to weight selection is shown by the fact that a popular solver as `CVX`, matlab software for disciplined convex programming [GB11], can only find the solution of (2.8) for networks with at most tens of thousands of links.

The optimal solution in larger networks can be found iteratively using a centralized subgradient method. The authors of [XB04] present a sub-gradient method for selecting weights on links in a network by minimizing the following unconstrained problem (whose solution is equivalent to solving problems (2.6) or (2.8)):

$$\underset{\mathbf{w}}{\text{argmin}} \;\; r(\mathbf{w}) = \rho(I - Q \times \text{diag}(\mathbf{w}) \times Q^T - G_1).$$

Each link weight is updated according to the following sub-gradient iteration:

$$w_l^{(k+1)} = w_l^{(k)} - \gamma^{(k)} g_l^{(k)} / ||\mathbf{g}^{(k)}||, \tag{2.9}$$

where $w_l^{(k)}$ is the weight on link l at iteration k, $g_l^{(k)}$ is the l-th component of a subgradient $\mathbf{g}^{(k)}$ of the objective function calculated in $\mathbf{w}^{(k)}$, and $\gamma^{(k)}$ is the stepsize satisfying the following sufficient conditions for convergence, $\lim_{k\to\infty} \gamma^{(k)} = 0$ and $\sum_{k=1}^{\infty} \gamma^{(k)} = \infty$. The components of the sub-gradient can be calculated as follows:

- if $r(\mathbf{w}) = \lambda_2(W)$, then

  $$g_l = -(u_i - u_j)^2, \text{ if } l \sim (i,j), \; l = 1, \ldots, m,$$

  where u_i is the i-th component of a unit eigenvector of the weight matrix $W(k)$ corresponding to the eigenvalue λ_2.

- if $r(\mathbf{w}) = -\lambda_n(W)$, then

  $$g_l = (u_i - u_j)^2, \text{ if } l \sim (i,j), \; l = 1, \ldots, m,$$

  where u_i is the i-th component of the unit eigenvector of the weight matrix $W(k)$ corresponding to the eigenvalue λ_n.

Contrary to the centralized approach for the subgradient method, in a distributed solution all the nodes in the network contribute to calculate the solution of the optimization problem. The whole network then benefits from nodes' processing capabilities. However, the subgradient approach given above is not distributed for different reasons. First, the stepsize used in (2.9) is normalized by $||\mathbf{g}^{(k)}||$ which cannot be locally computed by each node. While this problem can probably be circumvented by a different choice of the stepsize (without loosing the convergence properties of (2.9)), there are other aspects that make problematic this distributed

implementation. In fact this iterative procedure requires at every step to calculate $\lambda_2(W(k))$ and $\lambda_n(W(k))$, and determine an eigenvector corresponding to one of these two eigenvalues that is the largest in module. For the solution to be really distributed also these quantities have to be calculated in a distributed way. This is not an easy task. There are some distributed iterative techniques [KM04, FGGS09, YFG+08] that converge asymptotically to the correct eigenvalue-eigenvector pair, but then each step of the optimization procedure requires itself the convergence of an iterative sub-procedure to calculate the two eigenvalues and the corresponding eigenvectors with significant computation and communication costs. We remark in particular that at each step the sub-procedure has to run long enough to guarantee that the estimations are accurate enough to not jeopardize the convergence of the optimization procedure. Deciding when to terminate the sub-procedure at each step may require itself another distributed mechanisms or the use of worst-case bounds on the errors.

A similar optimization problem but with some additional constraints is to find the fastest converging algorithm for randomized gossiping, and it has been studied in [BGPS06]. The authors provide a subgradient method that projects the variables violating the constraints back onto the feasible set. The projection can be done in a distributed way and the stepsize sequence can be calculated at each node. Nevertheless, the gradient of the cost function depends also in this case on eigenvalues and eigenvectors of the underlying graph, so its calculation incurs the same problems exposed above.

Kim *et al.* in [KGP09] propose a weight selection algorithm using the qth-order spectral norm minimization (q-SNM). They showed that if a symmetric weight matrix is considered, then the solution of the q-SNM is equivalent to that of the symmetric FDLA problem. Nevertheless, their remark is not tailored for symmetric weight matrix because their algorithm is computationally more expensive than the SDP. Another global weight optimization to approximate problem (2.6) is given in [JXM10] where the authors consider a cost function over finite time horizon and observe numerically that the more eigenvalues are considered in the objective function $(\lambda_2, \lambda_3, \dots)$ the faster it is in the transient phase. In conclusion, how to solve the problem (2.6) in a distributed way is still an open challenge.

Some heuristics for the weight selection problem that guarantee convergence of the average protocol and attracted some interest in the literature either due to their distributed nature or to their easy implementation are the following [XBK07, XB04]:

- max degree weights (MD):
  $w_l = \frac{1}{\Delta+1} \quad \forall l = 1, \dots, m;$
- local degree (Metropolis) weights (LD):
  $w_l = \frac{1}{\max\{d_i, d_j\}+1} \; l \sim (i,j) \quad \forall l = 1, 2, \dots, m;$
- optimal constant weights (OC):
  $w_l = \frac{2}{\lambda_1(L)+\lambda_{n-1}(L)} \quad \forall l = 1, \dots, m;$

where $\Delta = \max_i\{d_i\}$ is the maximum degree in the network and L is the Laplacian of the graph. A similar heuristic, called neighborhood algorithm (NA) [AECN11], was proposed by the author in his master thesis [Cha11]. Each node i sets the weight of a link (i, j) depending on the similarity between its neighborhood set and the neighborhood of node j. NA quantifies such similarity by resorting to the Jaccard index defined in the set theory.[1]

2.3 Schatten Norm Minimization

We change the original minimization problem in (2.6) by considering a different cost function that is a monotonic function of the Schatten norm. The Schatten p-norm of a matrix W is the L_p-norm of its singular values, i.e., $||W||_{\sigma p} = (\sum_i \sigma_i^p)^{1/p}$. The minimization problem we propose is the following one:

$$\begin{aligned} \underset{W}{\text{argmin}} \quad & h(W) = ||W||_{\sigma p}^p \\ \text{subject to} \quad & W = W^T, \\ & W\mathbf{1} = \mathbf{1}, \\ & W \in \mathcal{C}_G, \end{aligned} \tag{2.10}$$

where p is an even positive integer. The following result establishes that (2.10) is a smooth convex optimization problem and also it provides an alternative expression of the cost function in terms of the trace of W^p. For this reason we refer to our problem also as *Trace Minimization* (TM).

Proposition 1. *For any even positive integer p, the function $h(W) = ||W||_{\sigma p}^p = Tr(W^p)$ is scalar-valued, smooth, and convex on its feasible domain when W is symmetric.*

Proof. We have $\text{Tr}(W^p) = \sum_{i=1}^n \lambda_i^p$. Since W is symmetric, its non-zero singular values are the absolute values of its non-zero eigenvalues [Mey00]. Given that p is even, then $\sum_{i=1}^n \lambda_i^p = \sum_{i=1}^n \sigma_i^p$. Therefore, $\text{Tr}(W^p) = ||W||_{\sigma p}^p$.

The Schatten norm $||W||_{\sigma p}$ is a nonnegative convex function, then h is convex because it is the composition of a non-decreasing convex function –the function x^p where x is non-negative– and a convex function [BV04, p. 84].

The function is also differentiable and we have [Ber05, p. 411]

$$\frac{\partial}{\partial w_{ij}} \text{Tr}(W^p) = p(W^{p-1})_{j,i} \quad . \tag{2.11}$$

□

We now illustrate the relation between (2.10) and the optimization (2.6). The following lemmas will prepare the result:

[1] For any two sets A, and B, the Jaccard index is: $J(A, B) = |A \cap B|/|A \cup B|$.

Lemma 1. *For any symmetric weight matrix W whose rows (and columns) sum to 1 and with eigevalues $\lambda_1(W) \geq \lambda_2(W) \geq \cdots \geq \lambda_n(W)$, there exist two integers $K_1 \in \{1, 2, \ldots n-1\}, K_2 \in \{0, 1, 2, \ldots n-1\}$ and a positive constant $\alpha < 1$ such that for any even positive integer p we have:*

$$1 + \tau(W)^p K_1 \leq Tr(W^p) \leq 1 + \tau(W)^p (K_1 + K_2 \alpha^p), \tag{2.12}$$

where

$$\tau(W) = \begin{cases} \rho(W) = \max\{\lambda_1(W), -\lambda_n(W)\} & \text{if } \rho(W) > 1, \\ \mu(W) = \max\{\lambda_2(W), -\lambda_n(W)\} & \text{if } \rho(W) \leq 1. \end{cases} \tag{2.13}$$

Proof. Let us consider the matrix W^2 and denote by $\nu_1, \nu_2, \ldots, \nu_r$ its distinct eigenvalues ordered by the largest to the smallest and by $m_1, m_2, \ldots, m_r$ their respective multiplicities. We observe that they are all non-negative and then they are also different in module. For convenience we consider $\nu_s = m_s = 0$ for $s > r$. Since p is an even positive integer, it can be written as $p = 2q$ where q positive integer. We can then write:

$$\mathrm{Tr}(W^p) = \sum_{i=1}^{n} \lambda_i^p = \sum_{i=1}^{r} m_i \nu_i^q.$$

The matrix W^2 has 1 as an eigenvalue. Let us denote by j its position in the ordered sequence of distinct eigenvalues, i.e., $\nu_j = 1$. Then it holds:

$$\mathrm{Tr}(W^p) = 1 + (m_j - 1) + \sum_{i \neq j} m_i \nu_i^q.$$

If $\rho(W) = 1$ (i.e., 1 is the largest eigenvalue in module of W), then 1 is also the largest eigenvalue of W^2 ($\nu_1 = 1$). If $m_1 > 1$, then it has to be either $\lambda_2(W) = 1$ (the multiplicity of the eigenvalue 1 for W is larger than 1) or $\lambda_n(W) = -1$. In both cases $\tau(W) = \mu(W) = 1$,

$$\mathrm{Tr}(W^p) = 1 + (m_1 - 1) + \sum_{i>1} m_i \nu_i^q$$

and the result holds with $K_1 = m_1 - 1$, $K_2 = \sum_{i>1} m_i$ and $\alpha = \sqrt{\nu_2}$. If $m_1 = 1$, then $\nu_2 = \lambda_2^2$. We can write:

$$\mathrm{Tr}(W^p) = 1 + \nu_2^q \left(m_2 + \sum_{i>2} m_i \left(\frac{\nu_i}{\nu_2} \right)^q \right)$$

and the result holds with $K_1 = m_2$, $K_2 = \sum_{i>2} m_i$, and $\alpha = \sqrt{\nu_3/\nu_2}$.

If $\rho(W) > 1$, then $\nu_1 = \rho(W)^2 > 1$ and we can write:

$$\mathrm{Tr}(W^p) = 1 + \nu_1^q \left(m_1 + \sum_{\substack{i>1 \\ i \neq j}} m_i \left(\frac{\nu_i}{\nu_1} \right)^q + (m_j - 1) \left(\frac{1}{\nu_1} \right)^q \right).$$

Then the result holds with $\tau(W) = \sqrt{\nu_1} = \rho(W)$, $K_1 = m_1$, $K_2 = \sum_{i>1} m_i$, and $\alpha = \sqrt{\nu_2/\nu_1}$. □

Lemma 2. *Let us denote by $W_{(p)}$ the solution of the minimization problem (2.10). If the graph of the network is strongly connected then $\tau\left(W_{(p)}\right) < 1$ for p sufficiently large.*

Proof. If the graph is strongly connected then there are multiple ways to assign the weights such that the convergence conditions (2.3)-(2.5) are satisfied. In particular the local degree method described in Section 2.2 is one of them. Let us denote by $W_{(LD)}$ its weight matrix. A consequence of the convergence conditions is that 1 is a simple eigenvalue of $W_{(LD)}$, and that all other eigenvalues are strictly less than one in magnitude [XB04]. It follows that $\tau\left(W_{(LD)}\right)$ in Lemma 1 is strictly smaller than one and that $\lim_{p\to\infty} \mathrm{Tr}\left(W_{(LD)}^p\right) = 1$. Then there exists a value p_0 such that for each $p > p_0$

$$\mathrm{Tr}\left(W_{(LD)}^p\right) < 2.$$

Let us consider the minimization problem (2.10) for a value $p > p_0$. $W_{(LD)}$ is a feasible solution for the problem, then

$$\mathrm{Tr}(W_{(p)}^p) \leq \mathrm{Tr}(W_{(LD)}^p) < 2.$$

Using this inequality and Lemma 1, we have:

$$1 + \tau\left(W_{(p)}\right)^p \leq 1 + \tau\left(W_{(p)}\right)^p K_1 \leq \mathrm{Tr}(W_{(p)}^p) < 2,$$

from which the lemma follows immediately. □

We are now ready to state our main results in the following two propositions:

Proposition 2. *If the graph of the network is strongly connected, then the solution $W_{(p)}$ of the Schatten Norm minimization problem (2.10) satisfies the consensus protocol convergence conditions for p sufficiently large, i.e.,*

$$W_{(p)} = W_{(p)}^T,\ W_{(p)}\mathbf{1} = \mathbf{1}, \quad \textit{and } \mu(W_{(p)}) < 1.$$

Proof. The solution of problem (2.10), $W_{(p)}$ is necessarily symmetric and its rows sum to 1. From Lemma 2 it follows that for p sufficiently large $\tau\left(W_{(p)}\right) < 1$ then by the definition of $\tau(.)$ it has to be $\rho(W_{(p)}) = 1$ and $\mu(W_{(p)}) < 1$. Therefore $W_{(p)}$ satisfies all the three convergence conditions (2.3)-(2.5) and then the consensus protocol converges.

It is further possible to show that in fact, as p approaches ∞, the Schatten Norm minimization problem (2.10) is equivalent to the minimization problem (2.6) (i.e., to minimize the second largest eigenvalue $\mu(W)$). To show this, we observe that with respect to the variable weight matrix W, minimizing $\mathrm{Tr}(W^p)$ is equivalent to minimizing $(\mathrm{Tr}(W^p) - 1)^{1/p}$. From Eq. (2.12), it follows:

$$\tau(W)K_1^{\frac{1}{p}} \leq (\mathrm{Tr}(W^p) - 1)^{\frac{1}{p}} \leq \tau(W)(K_1 + K_2\alpha^p)^{\frac{1}{p}}.$$

K_1 is bounded between 1 and $n-1$ and K_2 is bounded between 0 and $n-1$, and $\alpha < 1$,then it holds:

$$\tau(W)K_1^{\frac{1}{p}} \leq (\mathrm{Tr}(W^p) - 1)^{\frac{1}{p}} \leq \tau(W)K^{\frac{1}{p}},$$

with $K = 2(n-1)$. For p large enough $\tau\left(W_{(p)}\right) = \mu(W_{(p)})$, then

$$\left|(\mathrm{Tr}(W_{(p)}^p) - 1)^{\frac{1}{p}} - \mu(W_{(p)})\right| \leq \mu(W_{(p)})\left(K^{\frac{1}{p}} - 1\right) \leq K^{\frac{1}{p}} - 1.$$

Then the difference of the two cost functions converges to zero as p approaches infinity. □

Proposition 3. *The Schatten Norm minimization (2.10) is an approximation for the original problem (2.6) with a guaranteed error bound,*

$$|\mu(W_{(SDP)}) - \mu(W_{(p)})| \leq \mu(W_{(SDP)}) \times \epsilon(p),$$

where $\epsilon(p) = (n-1)^{1/p} - 1$ *and where* $W_{(SDP)}$ *and* $W_{(p)}$ *are the solutions of (2.6) and (2.10) respectively.*

Proof. Let S be the feasibility set of the problem (2.6) (and (2.10)), we have $\mu(W) = \max\{\lambda_2(W), -\lambda_n(W)\}$ and let $g(W) = \left(\sum_{i\geq 2}\lambda_i^p(W)\right)^{\frac{1}{p}}$. Since $W_{(SDP)}$ is a solution of (2.6), then

$$\mu(W_{(SDP)}) \leq \mu(W), \quad \forall W \in S. \tag{2.14}$$

Note that the minimization of $g(W)$ is equivalent to the minimization of $\mathrm{Tr}(W^p)$ when $W \in S$ (i.e., $\underset{W\in S}{\operatorname{argmin}}\, g(W) = \underset{W\in S}{\operatorname{argmin}}\, \mathrm{Tr}(W^p)$), then

$$g(W_{(p)}) \leq g(W), \quad \forall W \in S. \tag{2.15}$$

Finally for a vector $\mathbf{v} \in \mathbb{R}^m$ all norms are equivalent and in particular $||\mathbf{v}||_\infty \leq ||\mathbf{v}||_p \leq m^{1/p}||\mathbf{v}||_\infty$ for all $p \geq 1$. By applying this inequality to the vector whose elements are the $n-1$ eigenvalues different from 1 of the matrix W, we can write

$$\mu(W) \leq g(W) \leq (n-1)^{1/p}\mu(W), \quad \forall W \in S. \tag{2.16}$$

Using these three inequalities we can derive the desired bound:

$$\mu(W_{(SDP)}) \overset{(2.14)}{\leq} \mu(W_{(p)}) \overset{(2.16)}{\leq} g(W_{(p)}) \overset{(2.15)}{\leq} g(W_{(SDP)}) \overset{(2.16)}{\leq} (n-1)^{1/p}\mu(W_{(SDP)}), \tag{2.17}$$

where the number above the inequalities shows the equation used in deriving the bound. Therefore $\mu(W_{(SDP)}) \leq \mu(W_{(p)}) \leq (n-1)^{1/p}\mu(W_{(SDP)})$ and the thesis follows directly. □

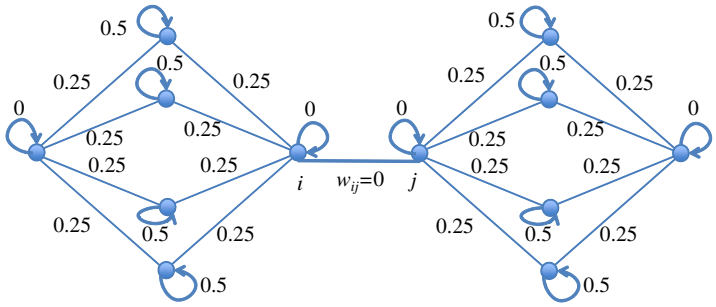

Figure 2.1: For this network the matrix solution of Schatten Norm minimization (2.10) with $p = 2$ does not guarantee convergence of average consensus to the true average because $w_{ij} = 0$ which separates the network into two parts, each of which can converge to a totally different value (but not to the average of initial values).

Remark: Comparing the results of Schatten Norm minimization (2.10) with the original problem (2.6), we observe that on some graphs the solution of problem (2.10) already for $p = 2$ gives the optimal solution of the main problem (2.6); this is for example the case for complete graphs.[2] However, on some other graphs, it may give a weight matrix that does not guarantee the convergence of the consensus protocol to the true average because the second largest eigenvalue can be larger than or equal to 1 (the other convergence conditions are intrinsically satisfied). We have built a toy example, shown in Fig. 2.1, where this happens. The solution of (2.10) assigns weight 0 to the link (i, j); $w_{ij} = 0$ separates the network into two disconnected subgraphs, so $\mu(W) = 1$ in this case. We know by Lemma 2 that this problem cannot occur for p large enough. In particular for the toy example the matrix solution for $p = 4$ already guarantees convergence. We discuss how to guarantee convergence for any value of p in Section 2.7.

Given that problem (2.10) is smooth and convex, it can be solved by interior point methods which would be a centralized solution. In the next section we are going to show a centralized approach using Newton's method, and in Section 2.5 a distributed algorithm using projected sub-gradients.

2.4 Newton's Method for Schatten Norm Minimization

Solutions of actual optimization problems are rarely expressed in a closed-form. More often they are obtained through iterative methods, that can be very effective in some cases (e.g., when the objective function is convex). Among the iterative ap-

[2] This can be easily checked. In fact, for any the matrix that guarantees the convergence of average consensus protocols it holds $\mu(W) \geq 0$ and $\mathrm{Tr}(W^2) \geq 1$ (because 1 is an eigenvalue of W). The matrix $\hat{W} = 1/n\mathbf{1}\mathbf{1}^T$ (corresponding to each link having the same weight $1/n$) has eigenvalues 1 and 0 with multiplicity 1 and $n-1$ respectively. Then $\mu(\hat{W}) = 0$ and $\mathrm{Tr}(\hat{W}^2) = 1$. It follows that $\hat{W}$ minimizes both the cost function of problem (2.6) and (2.10).

proaches, gradient methods converge under quite general hypotheses, but they may suffer from very slow convergence rates as they are coordinate dependent (scaling the variables in the problem affects the convergence speed). The Newton's method converges locally quadratically fast and is coordinate independent [BV04]. The drawback of Newton's method is that it requires the knowledge of the Hessian of the function that may be computationally too expensive to calculate. However, with the continuous increase of computation power and the existence of efficient algorithms for solving linear equations, Newton's method is the object of an increasing interest [WOEJ12, LSch, ARS12]. In this section, we provide a methodology for solving the Schatten norm minimization (2.10) using the Newton's method. We also show later by simulations that it significantly outperforms first order methods (decent gradient, Nestrov, etc.) both in terms of convergence speed and in term of robustness to the step size selection.

2.4.1 Preliminaries

The definitions of the gradient and Hessian of a scalar function vary depending on the argument of the function. For the scalar function of a *vector*, $f : \mathbb{R}^m \to \mathbb{R}$, the gradient of the function $f(\mathbf{x})$ with respect to the vector $\mathbf{x} \in \mathbb{R}^m$ is denoted by $\nabla_{\mathbf{x}} f \in \mathbb{R}^m$ and its Hessian is denoted by the matrix $\nabla^2_{\mathbf{x}} f \in \mathbb{R}^{m,m}$ whose elements are given by the following equations:

$$(\nabla_{\mathbf{x}} f)_l \triangleq \frac{\partial f}{\partial x_l}, \text{ and } \left(\nabla^2_{\mathbf{x}} f\right)_{l,k} \triangleq \frac{\partial^2 f}{\partial x_l \partial x_k} \quad \text{for } l,k = 1,\ldots,m.$$

For a scalar function of a *matrix*, $h : \mathbb{R}^{m,m} \to \mathbb{R}$, the gradient of the function $h(X)$ with respect to the vector $\text{vect}(X) \in \mathbb{R}^{m^2,1}$ is denoted by $\nabla_X h \in \mathbb{R}^{m^2,1}$ and its Hessian is denoted by the matrix $\nabla^2_X h \in \mathbb{R}^{m^2,m^2}$ whose elements are given by the equations:

$$\nabla_X h_{(ij)} \triangleq \frac{\partial h}{\partial x_{ij}}, \text{ and } \nabla^2_X h_{(ij)(st)} \triangleq \frac{\partial^2 h}{\partial x_{ij} \partial x_{st}}.$$

Newton's method is an iterative technique that finds the roots of a function. For an unconstrained convex minimization problem, the roots of the gradient of the function to minimize are the minimizers of the function itself. The Newton's method is very popular due to its fast speed of convergence. Consider the following unconstrained minimization problem:

$$\text{minimize } f(\mathbf{w}), \tag{2.18}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is strongly convex and twice continuously differentiable. We suppose that the problem has a solution f^* and the solution is obtained at $\mathbf{w}^*$, i.e., $f^* = f(\mathbf{w}^*)$. Since f is a convex and differentiable function, a point $\mathbf{w}^*$ is optimal if and only if the gradient of the function vanishes:

$$\nabla_{\mathbf{w}} f(\mathbf{w}^*) = \mathbf{0}. \tag{2.19}$$

Therefore, solving the m equations of m variables in (2.19) is equivalent to solving the optimization problem (2.18). The Newton's method (also called damped Newton's method) is outlined below [BV04]:

Newton's Method Algorithm

Given

A starting point $\mathbf{w} \in \mathbf{dom} f$, a tolerance $\epsilon > 0$.

Repeat

1. Compute Newton's step and decrement:

$$\Delta \mathbf{w} := \left(\nabla^2_{\mathbf{w}} f(\mathbf{w})\right)^{-1} \nabla_{\mathbf{w}} f(\mathbf{w}),$$

$$\delta^2 := \nabla_{\mathbf{w}} f(\mathbf{w})^T \left(\nabla^2_{\mathbf{w}} f(\mathbf{w})\right)^{-1} \nabla_{\mathbf{w}} f(\mathbf{w}).$$

2. Stopping criterion: if $\delta^2/2 \leq \epsilon$ exit.
3. Line search: use exact or backtracking line search to find t.
4. Update:

$$\mathbf{w} := \mathbf{w} - t\Delta \mathbf{w}.$$

In the following, we will apply the Newton's method to the Schatten norm minimization problem (2.10).

2.4.2 The Unconstrained Minimization

As mentioned earlier, the constrains in (2.10) lead to the possibility to write W as follows: $W = I - Q \times \text{diag}(\mathbf{w}) \times Q^T$, where $\mathbf{w} \in \mathbb{R}^m$ is the vector of all the weight links w_l, $l = 1, \ldots, m$. It follows that Schatten Norm minimization (2.10) is equivalent to the following unconstrained problem:

$$\text{minimize } f(\mathbf{w}) = \text{Tr}\left((I - Q \times \text{diag}(\mathbf{w}) \times Q^T)^p\right). \tag{2.20}$$

2.4.3 Gradient and Hessian

To apply Newton's method to minimize the function f in (2.20), we have to calculate first the gradient $\nabla_{\mathbf{w}} f$ and the Hessian matrix $\nabla^2_{\mathbf{w}} f$. The function f is a composition of the scalar function $h(W) = \text{Tr}(W^p)$ and the matrix function $W = I - Q\text{diag}(\mathbf{w})Q^T$:

$$f(\mathbf{w}) = \text{Tr}(W^p)|_{W = I_n - Q\text{diag}(\mathbf{w})Q^T}.$$

For the gradient $\nabla_{\mathbf{w}} f$, it holds for $l = 1, \ldots, m$:

$$(\nabla_{\mathbf{w}} f)_l = \sum_{i,j \in V} \nabla_W h_{(ij)} \frac{\partial w_{ij}}{\partial w_l},$$

where $\nabla_W h_{(ij)} = p(W^{p-1})_{j,i}$ (from (2.11)). Due to the conditions mentioned earlier ($w_{ij} = w_{ji} = w_l$ for all $l \sim (ij)$ and $w_{ij} = 0$ if $(ij) \notin E$ and $w_{ii} = 1 - \sum_{j \in N_i} w_{ij}$), if $l \sim (ab)$ we have

$$\frac{\partial w_{ij}}{\partial w_l} = \begin{cases} +1 & \text{if } i = a \text{ and } j = b \\ +1 & \text{if } i = b \text{ and } j = a \\ -1 & \text{if } i = a \text{ and } j = a \\ -1 & \text{if } i = b \text{ and } j = b \\ 0 & \text{else.} \end{cases} \tag{2.21}$$

We can then calculate the gradient $\nabla_{\mathbf{w}} f \in \mathbb{R}^m$. In particular for $l \sim (ab)$ we have,

$$\begin{aligned} (\nabla_{\mathbf{w}} f)_l &= \nabla_W h_{(ab)} + \nabla_W h_{(ba)} - \nabla_W h_{(aa)} - \nabla_W h_{(bb)} \\ &= p(W^{p-1})_{b,a} + p(W^{p-1})_{a,b} - p(W^{p-1})_{a,a} - p(W^{p-1})_{b,b}. \end{aligned} \tag{2.22}$$

Applying the chain rule for the Hessian and considering directly that all the second order derivatives like $\frac{\partial^2 w_{ij}}{\partial w_l \partial w_k}$ are null because the mapping is a linear transformation, we obtain that for $l, k = 1, \ldots, m$:

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = \frac{\partial^2 f}{\partial w_l \partial w_k} = \sum_{i,j,s,t} \nabla^2_W h_{(ij)(st)} \frac{\partial w_{ij}}{\partial w_l} \frac{\partial w_{st}}{\partial w_k}. \tag{2.23}$$

For the calculation of the Hessian of f, let us first give the expression of $\nabla^2_W h_{(ij)(st)}$. Notice that for any a and b we have $\frac{\partial w_{ab}}{\partial w_{st}} = \delta_{as}\delta_{tb}$, where δ_{uv} is the Kronecker delta, i.e., $\delta_{uv} = 1$ if $u = v$, $\delta_{uv} = 0$ otherwise. Then the Hessian of $h(W)$ is given by:

$$\begin{aligned} \nabla^2_W h_{(ij)(st)} &= \frac{\partial^2 \mathrm{Tr}(W^p)}{\partial w_{ij} \partial w_{st}} \\ &= \frac{\partial}{\partial w_{st}} \left(p(W^{p-1})_{j,i}\right) \\ &= p \frac{\partial}{\partial w_{st}} \sum_{u_1, u_2, \ldots, u_{p-2}} w_{ju_1} w_{u_1u_2} w_{u_2u_3} \ldots w_{u_{p-2}i} \\ &= p \frac{\partial}{\partial w_{st}} \sum_{u_1, u_2, \ldots, u_{p-2}} \delta_{js}\delta_{tu_1} w_{u_1u_2} w_{u_2u_3} \ldots w_{u_{p-2}i} \\ &\quad + p \frac{\partial}{\partial w_{st}} \sum_{u_1, u_2, \ldots, u_{p-2}} w_{ju_1} \delta_{u_1s}\delta_{tu_2} w_{u_2u_3} \ldots w_{u_{p-2}i} \\ &\quad + \cdots + p \frac{\partial}{\partial w_{st}} \sum_{u_1, u_2, \ldots, u_{p-2}} w_{ju_1} w_{u_1u_2} w_{u_2u_3} \ldots \delta_{u_{p-2}s}\delta_{ti} \\ &= p \sum_{z=0}^{p-2} (W^z)_{j,s} (W^{p-2-z})_{t,i}. \end{aligned} \tag{2.24}$$

Thus for the calculation of the Hessian of f, let $l \sim (ab)$, $k \sim (cd)$ be given links. Only 16 of the m^4 terms in Eq. (2.23) (those corresponding to $i, j \in \{a, b\}$

and $s, t \in \{c, d\}$) are different from zero because of (2.21). Moreover using the expression of $\nabla^2_X h_{(ij)(st)}$ in (2.24) and grouping the terms, we obtain the compact form:

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = p \sum_{z=0}^{p-2} \psi(z)\psi(p-2-z), \tag{2.25}$$

where

$$\psi(z) = (W^z)_{a,c} + (W^z)_{b,d} - (W^z)_{a,d} - (W^z)_{b,c}.$$

2.4.4 Newton's Direction $\Delta\mathbf{w}$

Let $\mathbf{g} \in \mathbb{R}^m$ and $H \in \mathbb{R}^{m\times m}$ such that $\mathbf{g} = \nabla_{\mathbf{w}} f(\mathbf{w})$ whose elements are given by equation (2.22) and $H = \nabla^2_{\mathbf{w}} f(\mathbf{w})$ whose elements are given by equation (2.25). Then the direction $\Delta\mathbf{w}$ to update the solution in Newton's method can be obtained solving the linear system $H\Delta\mathbf{w} = \mathbf{g}$.

2.4.5 Line Search

The Newton's method uses *exact line search* if at each iteration the stepsize is selected in order to guarantee the maximum amount of decrease of the function f in the descent direction, i.e., t is selected as the global minimizer of the univariate function $\phi(t)$:

$$\phi(t) = f(\mathbf{w} - t\Delta\mathbf{w}),\ t > 0.$$

Usually exact line search is very difficult to implement, possible alternatives can be the *pure* Newton's method that selects a stepsize $t = 1$ at every iteration or the *backtracking line search* if t is selected to guarantee some sufficient amount of decrease in the function $\phi(t)$. But we benefit from the convexity of our problem to derive a procedure which gives a high precision estimate of the optimal choice of the exact line search stepsize. Notice that $\phi(t)$ can be written as follows:

$$\begin{aligned}
\phi(t) &= f(\mathbf{w} - t\Delta\mathbf{w}) \\
&= \mathrm{Tr}((I_n - Q\mathrm{diag}(\mathbf{w} - t\Delta\mathbf{w})Q^T)^p) \\
&= \mathrm{Tr}((I_n - Q\mathrm{diag}(\mathbf{w})Q^T + tQ\mathrm{diag}(\Delta\mathbf{w})Q^T)^p) \\
&= \mathrm{Tr}((W + tU)^p) \\
&= h(W + tU),
\end{aligned}$$

where $U = Q\mathrm{diag}(\Delta\mathbf{w})Q^T$ is a symmetric matrix. Since (2.10) is a smooth convex optimization problem, h is also smooth and convex when restricted to any line that intersects its domain. Then $\phi(t) = h(W + tU)$ is convex in t and applying the chain rule to the composition of the function $h(Y) = \mathrm{Tr}(Y^p)$ and $Y(t) = W + tU$ (similarly to what we have done for f in (2.23)), we can find the first and second derivative:

$$\phi'(t) = \sum_{i,j} \frac{\partial h}{\partial y_{ij}} u_{ij} = p \sum_i (Y^{p-1}U)_{i,i} = p\mathrm{Tr}(Y^{p-1}U),$$

$$\phi''(t) = \frac{\mathrm{d}\phi'(t)}{\mathrm{d}t} = p \times \mathrm{Tr}\left(\sum_{q=0}^{p-2} Y^{p-2-q} U Y^q U\right).$$

So we can apply a basic Newton's method to find the optimal t:

Let $t_1 = 1$ and $t_0 = 0$, select a tolerance $\eta > 0$,
$n := 1$;
while $|t_n - t_{n-1}| > \eta$
$t_{n+1} \leftarrow t_n - \frac{\phi'(t_n)}{\phi''(t_n)}$;
$n := n + 1$;
end while

At the end of this procedure, we select $t = t_n$ to be used as the stepsize of the iteration.

2.4.6 The Algorithm

We summarize the Newton's method used for the trace minimization problem (2.10):

Step 0: Choose a weight matrix $W^{(0)}$ that satisfies the conditions given in (2.10) (e.g., I_n is a feasible starting weight matrix). Choose a precision ϵ and set $k \leftarrow 0$.

Step 1: Calculate $\nabla_{\mathbf{w}} f^{(k)}$ from equation (2.22) (call this gradient $\mathbf{g}$).

Step 2: Calculate $\nabla^2_{\mathbf{w}} f^{(k)}$ from equation (2.25) (since f is a convex function, we have $\nabla^2_{\mathbf{w}} f^{(k)}$ is a semi-definite positive matrix, let $H = \nabla^2_{\mathbf{w}} f^{(k)} + \gamma I_m$ where γ can be chosen to be the machine precision to guarantee that H is positive definite and thus can have an inverse H^{-1}).

Step 3: Calculate Newton's direction $\Delta\mathbf{w}^{(k)} = H^{-1}\mathbf{g}$. **Stop** if $||\Delta\mathbf{w}^{(k)}|| \leq \epsilon$.

Step 4: Use the exact line search to find the stepsize $t^{(k)}$.

Step 5: Update the weight matrix by the following equation:

$$W^{(k+1)} = W^{(k)} + t^{(k)} Q \mathrm{diag}(\Delta\mathbf{w}^{(k)}) Q^T.$$

Step 6: Increment iteration $k \leftarrow k + 1$. Go to **Step 1**.

2.4.7 Closed Form Solution for $p = 2$

Interestingly, for $p = 2$ the Newton's method converges in 1 iteration. In fact for $p = 2$, the problem (2.10) is the following:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & h(W) = \mathrm{Tr}(W^2) = \sum_{i,j} w_{ij}^2 \\ \text{subject to} \quad & W = W^T, W\mathbf{1}_n = \mathbf{1}_n, W \in \mathcal{C}_G. \end{aligned} \tag{2.26}$$

Theorem 1. *Let $W_{(2)}$ be the solution of the optimization problem (2.26), then we have:*

$$W_{(2)} = I_n - Q\,diag\left((I_m + \frac{1}{2}Q^TQ)^{-1}\mathbf{1}_m\right)Q^T, \tag{2.27}$$

where Q is the incidence matrix of the graph G.

Proof. The optimization function is quadratic in the variables w_{ij}, so applying Newton's algorithm to minimize the function gives convergence in one iteration independent from the initial starting point $W^{(0)}$. Let $W^{(0)} = I_n$ which is a feasible initial starting point. The gradient $\mathbf{g}$ can be calculated according to equation (2.22):

$$\begin{aligned} g_l &= 2\left((I_n)_{i,j} + (I_n)_{j,i} - (I_n)_{i,i} - (I_n)_{j,j}\right) \\ &= 2(0 + 0 - 1 - 1) = -4 \quad \forall l = 1, \ldots, m, \end{aligned}$$

so in vector form $\mathbf{g} = -4 \times \mathbf{1}_m$. To calculate the Hessian $\nabla^2_W f$, we apply equation (2.25) for $p = 2$, so for any two links $l \sim (ab)$ and $k \sim (cd)$, we have

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = 2 \times \left((I_n)_{a,c} + (I_n)_{b,d} - (I_n)_{a,c} - (I_n)_{b,d}\right)^2,$$

and thus

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = \begin{cases} 2 \times (2)^2 & \text{if } l = k \\ 2 \times (1)^2 & \text{if } l \text{ and } k \text{ share a common vertex,} \\ 0 & \text{else.} \end{cases} \tag{2.28}$$

In matrix form, we can write the Hessian as follows:

$$\nabla^2_{\mathbf{w}} f = 2 \times (2I_m + Q^TQ),$$

where Q is the incidence matrix of the graph given earlier (in fact, $Q^TQ - 2I_m$ is the adjacency matrix of what is called the line graph of G). Notice that since Q^TQ is semi-definite positive all the eigenvalues of the Hessian are larger than 2 and then the Hessian is invertible. The Newton's direction is calculated as follows:

$$\Delta\mathbf{w} = H^{-1}\mathbf{g} = -(I_m + \frac{1}{2}Q^TQ)^{-1}\mathbf{1}_m.$$

Thus the optimal solution for the problem for $p = 2$ is:

$$\begin{aligned} W_{(2)} &= W^{(0)} + Q\text{diag}(\Delta\mathbf{w})Q^T \\ &= I_n - Q\text{diag}\left((I_m + \frac{1}{2}Q^TQ)^{-1}\mathbf{1}_m\right)Q^T. \end{aligned}$$

□

If the graph is D-regular, the previous expression further simplifies. A D-regular graph is a graph where every node has the same number of neighbors which is D. Examples of D regular graphs are cycles (2-regular) and the complete graph ($n-1$-regular).

In fact, the sum of any row in the matrix Q^TQ is equal to $2D$, then $2D$ is an eigenvalue that corresponds to the eigenvector $\mathbf{1}_m$. Since Q^TQ is a symmetric matrix, it has an eigenvalue decomposition form:

$$Q^TQ = \sum_k \lambda_k \mathbf{v}_k \mathbf{v}_k^T,$$

where $\{\mathbf{v}_k\}$ is an orthonormal set of eigenvectors (without loss of generality, let $\mathbf{v}_1 = \frac{1}{\sqrt{m}}\mathbf{1}_m$). Moreover, $(I_m + \frac{1}{2}Q^TQ)$ is invertible because it is positive definite and has the same eigenvectors as Q^TQ. Considering its inverse as a function of Q^TQ, we can write:

$$(I_m + \frac{1}{2}Q^TQ)^{-1} = \sum_k (1 + \frac{\lambda_k}{2})^{-1} \mathbf{v}_k \mathbf{v}_k^T.$$

Since $\mathbf{1}_m$ is an eigenvector of Q^TQ and therefore of $(I_m + \frac{1}{2}Q^TQ)^{-1}$, it is perpendicular to all the others ($\mathbf{v}_k^T\mathbf{1}_m = 0$ for all $k \neq 1$). Hence, it follows that:

$$(I_m + \frac{1}{2}Q^TQ)^{-1}\mathbf{1}_m = (1 + \frac{\lambda_1}{2})^{-1}\mathbf{v}_1(\frac{m}{\sqrt{m}}) = \frac{1}{1+D}\mathbf{1}_m.$$

As a result, the solution of the optimization is given by,

$$W_{(2)} = I_n - \frac{1}{1+D}QQ^T,$$

or equivalently as function of $\mathbf{w}$:

$$w_l = \frac{1}{1+D} \ \forall l = 1, \ldots, m.$$

Interestingly, the solution of the suggested optimization problem for $p = 2$ gives the same matrix on D-regular graphs as other weight selection algorithms for average consensus as Metropolis weight selection or maximum degree weight selection.

2.5 A Distributed Algorithm for Schatten Norm Minimization

In this section we will show that the optimization problem (2.10) can be solved in a distributed way using gradient methods. By distributed algorithm we mean an algorithm where each node only needs to retrieve information from a limited neighborhood (possibly larger than N_i) in order to calculate the weights on its incident links.

We have already seen that W can be written as follows: $W = I - Q \times \mathrm{diag}(\mathbf{w}) \times Q^T$, where $\mathbf{w} \in \mathbb{R}^m$ is the vector of all the weight links w_l, $l = 1, \ldots, m$. It follows that Schatten Norm minimization (2.10) is equivalent to the following unconstrained problem:

$$\text{minimize } f(\mathbf{w}) = \mathrm{Tr}\left((I - Q \times \mathrm{diag}(\mathbf{w}) \times Q^T)^p\right). \tag{2.29}$$

We will give a distributed algorithm to solve the Schatten Norm minimization (2.10) by applying gradient techniques to problem (2.29). Since the cost function to optimize is smooth and convex as we proved in Proposition 1, if the gradient technique converges to a stationary point, then it converges to the global optimum. The gradient method uses the simple iteration:

$$w_l^{(k+1)} = w_l^{(k)} - \gamma^{(k)} g_l^{(k)} \ \forall l = 1, \dots, m \ ,$$

where $\gamma^{(k)}$ is the stepsize at iteration k and $g_l^{(k)}$ is the l-th component of the gradient $\mathbf{g}^{(k)}$ of the function $f(\mathbf{w})$. At every iteration k, starting from a feasible solution for link weights, $w_l^{(k)}$, we calculate the gradient $g_l^{(k)}$ for every link, and then we obtain a new weight value $w_l^{(k+1)}$.

There are different conditions on the function $f(.)$ and on the stepsize sequence that can guarantee convergence. A distributed computational model for optimizing a sum of non-smooth convex functions is proposed in [NO09, LO11] and its convergence is proved for bounded (sub)gradients for different network dynamics. For a similar objective function, the authors in [JKJJ08] study the convergence of a projected (sub)-gradient method with constant stepsize. For unbounded gradients, the algorithm in [Pol87, Section 5.3.2, p. 140] guarantees global convergence but requires a centralized calculation of the stepsize sequence. Because the objective function in (2.29) has unbounded gradient, our distributed implementation combines ideas from unbounded gradients methods and the projecting methods using theorems from [BNO03]. In particular, we will add a further constraint to (2.29), looking for a solution in a compact set X, and we will consider the following projected gradient method:

$$\mathbf{w}^{(k+1)} = P_X \left(\mathbf{w}^{(k)} - \gamma^{(k)} \mathbf{g}^{(k)} \right) ,$$

where $P_X()$ is the projection on the set X. We can show that by a particular choice of X and $\gamma^{(k)}$ the method converges to the solution of the original problem. Moreover, all the calculations can be performed in a distributed way on the basis of local knowledge. In particular, we will show that:

- nodes incident to l are able to calculate $g_l^{(k)}$ using only information they can retrieve from their (possibly extended) neighborhood;
- the stepsize sequence $\gamma^{(k)}$ is determined a priori and then nodes do not need to evaluate the function f or any other global quantity to calculate it;
- the projection on set X can be performed component-wise, and locally at each node;
- the global convergence of the projected gradient method is guaranteed.

We will start by g_l and show that it only depends on information local to nodes i and j incident to the link $l \sim (i, j)$, then we will discuss the choice of the stepsize $\gamma^{(k)}$ and of the projection set X.

2.5.1 Locally Computed Gradient

The gradient g_l of the function $f(\mathbf{w})$ for $l \sim (i,j)$ can be calculated following equation (2.22):

$$\begin{aligned} g_l &= \frac{\partial f(\mathbf{w})}{\partial w_l} \\ &= p\left(\left(W^{p-1}\right)_{ji} + \left(W^{p-1}\right)_{ij} - \left(W^{p-1}\right)_{ii} - \left(W^{p-1}\right)_{jj}\right). \end{aligned} \tag{2.30}$$

It is well know from graph theory that if we consider W to be the adjacency matrix of a weighted graph G, then $(W^s)_{ij}$ is a function of the weights on the edges of the $i-j$ walks (i.e., the walks from i to j) of length exactly s (in particular if A is the adjacency matrix of an unweighted graph, then $(A^s)_{ij}$ is the number of distinct $i-j$ s-walks [Wes00]). Since for a given p the gradient g_l, $l \sim (i,j)$, depends on the $\{ii, jj, ij, ji\}$ terms of the matrix W^{p-1}, g_l can be calculated locally by using only the weights of links and nodes at most $\frac{p}{2}$ hops away from i or j.[3] Practically speaking, at each step, nodes i and j need to contact all the nodes up to $p/2$ hops away in order to retrieve the current values of the weights on the links of these nodes and the values of weights on the nodes themselves. For example, when $p = 2$, then the minimization is the same as the minimization of the Frobenius norm of W since $\mathrm{Tr}(W^2) = \sum_{i,j} w_{ij}^2 = ||W||_F^2$, and the gradient g_l can be calculated as $g_l = 2\times(2W_{ij} - W_{ii} - W_{jj})$ which depends only on the weights of the vertices incident to that link and the weight of the link itself. More details about the operations to carry and their cost in Section 2.5.3.

An advantage of our approach is that it provides a trade-off between locality and optimality. In fact, the larger the parameter p, the better the solution of problem (2.10) approximates the solution of problem (2.6), but at the same time the larger is the neighborhood from which each node needs to retrieve the information. When $p = 2$, g_l where $l \sim (i,j)$ only depends on the weights of subgraph induced by the two nodes i and j. For $p = 4$, the gradient g_l depends only on the weights found on the subgraph induced by the set of vertices $N_i \cup N_j$, then it is sufficient that nodes i and j exchange the weights of all their incident links.

2.5.2 Choice of Stepsize and Projection set

The global convergence of gradient methods (i.e., for any initial condition) has been proved under a variety of different hypotheses on the function f to minimize and on the step size sequence $\gamma^{(k)}$. In many cases the step size has to be adaptively selected on the basis of the value of the function or of the module of its gradient at the current estimate, but this cannot be done in a distributed way for the function $f(\mathbf{w})$. This leads us to look for convergence results where the step size sequence can be fixed ahead of time. Moreover the usual conditions, like Lipschitzianity or boundness of the gradient, are not satisfied by the function $f(.)$ over all the feasible

[3] If a link or a node is more than $p/2$ hops away both from node i and node j, then it cannot belong to a $i-j$ walk of length p.

set. For this reason we add another constraint to our original problem (2.29) by considering that the solution has to belong to a given convex and compact set X. Before further specifying how we choose the set X, we state our convergence result.

Proposition 4. *Given the following problem*

$$\begin{aligned} &\textit{minimize} && f(\mathbf{w}) = Tr\left((I - Q \times diag(\mathbf{w}) \times Q^T)^p\right), \\ &\textit{subject to} && \mathbf{w} \in X \end{aligned} \tag{2.31}$$

where $X \subseteq \mathbb{R}^m$ is a convex and compact set, if $\sum_k \gamma^{(k)} = \infty$ and $\sum_k \left(\gamma^{(k)}\right)^2 < \infty$, then the following iterative procedure converges to the minimum of f in X:

$$\mathbf{w}^{(k+1)} = P_X\left(\mathbf{w}^{(k)} - \gamma^{(k)}\mathbf{g}^{(k)}\right), \tag{2.32}$$

where $P_X(.)$ is the projection operator on the set X and $\mathbf{g}^{(k)}$ is the gradient of f evaluated in $\mathbf{w}^{(k)}$.

Proof. The function f is continuous on a compact set X, so it has a point of minimum. Moreover also the gradient $\mathbf{g}$ is continuous and then bounded on X. The result then follows from Proposition 8.2.6 in [BNO03, pp. 480]. □

For example, $\gamma^{(k)} = a/(b+k)$ where $a > 0$ and $b \geq 0$ satisfies the step size condition in Proposition 4.

While the convergence is guaranteed for any set X convex and compact, we have two other requirements. First, it should be possible to calculate the projection P_X in a distributed way. Second, the set X should contain the solution of the optimization problem (2.20). About the first issue, we observe that if X is the cartesian product of real intervals, i.e., if $X = [a_1, b_1] \times [a_2, b_2] \times \ldots [a_m, b_m]$, then we have that the l-th component of the projection on X of a vector $\mathbf{y}$ is simply the projection of the l-th component of the vector on the interval $[a_l, b_l]$, i.e.,

$$[P_X(\mathbf{y})]_l = P_{[a_l,b_l]}(y_l) = \begin{cases} a_l & \text{if } y_l < a_l, \\ y_l & \text{if } a_l \leq y_l \leq b_l, \\ b_l & \text{if } b_l < y_l. \end{cases} \tag{2.33}$$

Then in this case Eq. (2.32) can be written component-wise as

$$w_l^{(k+1)} = P_{[a_l,b_l]}(w_l^{(k)} - \gamma^{(k)} g_l^{(k)}).$$

We have shown in the previous section that g_l can be calculated in a distributed way, then the iterative procedure can be distributed. About the second issue, we choose X so that it includes all the weight matrices with spectral radius at most 1. The following lemma suggests a possible choice of X.

Lemma 3. *Let W be a real and symmetric matrix where each row (and column) sums to 1, then the following holds,*

$$\rho(W) = 1 \implies \max_{i,j} |w_{ij}| \leq 1.$$

Proof. Since W is real and symmetric, then we can write W as follows

$$W = S\Lambda S^T,$$

where S is an orthonormal matrix ($S^TS = SS^T = I$), and Λ is a diagonal matrix having $\Lambda_{kk} = \lambda_k$ and λ_k is the k-th largest eigenvalue of W. Let $\mathbf{r}_k$ and $\mathbf{c}_k$ be the rows and columns of S respectively and $r_k^{(i)}$ be the i-th element of this vector. So,

$$W = \sum_k \lambda_k \mathbf{c}_k \mathbf{c}_k^T,$$

and

$$\begin{aligned} |w_{ij}| &= |\sum_k \lambda_k c_k^{(i)} c_k^{(j)}| && (2.34)\\ &\leq \sum_k |c_k^{(i)}||c_k^{(j)}| && (2.35)\\ &= \sum_k |r_i^{(k)}||r_j^{(k)}| && (2.36)\\ &\leq ||\mathbf{r}_i||_2 ||\mathbf{r}_j||_2 && (2.37)\\ &= 1. && (2.38) \end{aligned}$$

The transition from (2.34) to (2.35) is due to the fact $\rho(W) = 1$, the transition from (2.36) to (2.37) is due to Cauchy–Schwarz inequality. The transition from (2.37) to (2.38) is due to the fact that S is an orthonormal matrix. □

A consequence of Lemma 3 is that if we choose $X = [-1, 1]^m$ the weight vector of the matrix solution of problem (2.6) necessarily belongs to X (the weight matrix satisfies the convergence conditions). The same is true for the solution of problem (2.20) for p large enough because of Proposition 2. The following proposition summarizes our results.

Proposition 5. *If the graph of the network is strongly connected, then the following distributed algorithm converges to the solution of the Schatten norm minimization problem for p large enough:*

$$w_l^{(k+1)} = P_{[-1,1]}(w_l^{(k)} - \gamma^{(k)} g_l^{(k)}), \quad \forall l = 1, \ldots, m, \tag{2.39}$$

where $\sum_k \gamma^{(k)} = \infty$ and $\sum_k \left(\gamma^{(k)}\right)^2 < \infty$.

Proof. The set $X = [-1, 1]^m$, on which the gradient iterate is projected, is a convex and compact set. From Proposition 2, for p sufficiently large we have $\mu(W_{(p)}) < 1$ and hence $\rho(W_{(p)}) = 1$. Then by applying Lemma 3, the weight matrix $W_{(p)}$ has necessarily link weights in X. Therefore, since the solution of the Schatten norm minimization problem for p large enough lies in X, Proposition 4 ends the proof. □

Remark: The stepsize $\gamma^{(k)}$ in Propositions 4 and 5 can be replaced by a constant stepsize (i.e., $\gamma^{(k)} = \gamma$ for all k) and the convergence results will still hold provided that γ is small enough ($0 < \gamma < 2/K$ where K is the Lipschitz constant of the gradient of f on X, see Theorem 1 in [Pol87, p. 207]). The advantage of a constant stepsize is that it provides better rate of convergence (the convergence can be with the rate of geometric progression when the function is strongly convex), but the nodes should be able to know K (or at least an upped bound).

2.5.3 Complexity of the Algorithm

Our distributed algorithm for Schatten Norm minimization requires to calculate at every iteration, the stepsize $\gamma^{(k)}$, the gradient $g_l^{(k)}$ for every link, and a projection on the feasible set X. Its complexity is determined by the calculation of link gradient g_l, while the cost of the other operations is negligible. In what follows, we detail the computational costs (in terms of number of operations and memory requirements) and communication costs (in terms of volume of information to transmit) incurred by each node for the optimization with the two values $p = 2$ and $p = 4$.

2.5.3.1 Complexity for $p = 2$

For $p = 2$, Eq. (2.30) reduces to $g_l = 2 \times (2W_{ij} - W_{ii} - W_{jj})$. Nodes are aware of their own weights (W_{ii}) and of the weights of the links they are incident to (W_{ij}), hence the only missing parameter in the equation is their neighbors self weight (W_{jj}). So at every iteration of the subgradient method, nodes must broadcast their self weight to their neighbors. We can say that the computational complexity for $p = 2$ is negligible and the communication complexity is 1 message carrying a single real value (w_{ii}) per link, per node and per iteration.

2.5.3.2 Complexity for $p = 4$

For $p = 4$, the node must collect information from a larger neighborhood. The gradient at link $l \sim (i, j)$ is given by $g_l = 4\big((W^3)_{ij} + (W^3)_{ji} - (W^3)_{ii} - (W^3)_{jj}\big)$. From the equation of g_l it seems like the node must be aware of all the weight matrix in order to calculate the 4 terms in the equation, however this is not true. As hinted in the previous section, each of the 4 terms can be calculated only locally from the weights within 2-hops from i or j. In fact, $(W^3)_{ij}$ depends only on the weights of links covered by a walk with 3 jumps: starting from i the first jump reaches a neighbor of i, the second one a neighbor of j and finally the third jump finishes at j, then we cannot move farther than 2 hops from i. Then $(W^3)_{ij}$ can be calculated at node i as follows: every node s in N_i, sends its weight vector $\mathbf{W}_s$ to i, where $\mathbf{W}_s \in \mathbb{R}^{|N_s|}$ is a vector that contains all weights selected by the node s to its neighbors, i.e., the weights $\{w_{st}, t \in N_s\}$. The same is true for the addend $(W^3)_{ji}$. The term $(W^3)_{ii}$ depends on the walks of length 3 starting and finishing in i, then node i can calculate it once it knows $\mathbf{W}_s$ for each s in N_i. Finally, the calculation of the term $(W^3)_{jj}$ at node i requires i to know more information about the links

existing among the neighbors of node j. Instead of the transmission of this detailed information, we observe that node j can calculate the value $(W^3)_{jj}$ (as node i can calculate $(W^3)_{ii}$) and then can transmit directly the result of the calculation to node i. Therefore, the calculation of g_l by node i for every link l incident to i can be done in three steps:

1. Create the subgraph H_i containing the neighbors of i and the neighbors of its neighbors by sending $(\mathbf{W}_i)$ and receiving the weight vectors $(\mathbf{W}_s)$ from every neighbor s.

2. Calculate $(W^3)_{ii}$ and broadcast it to the neighbors (and receive $(W^3)_{ss}$ from every neighbor s).

3. Calculate g_l.

We evaluate now both the computational and the communication complexity.

- Computation Complexity: Each node i must store the subgraph H_i of its neighborhood. The number of nodes of H_i is $n_H \leq \Delta^2 + 1$, the number of links of H_i is $m_H \leq \Delta^2$ where Δ is the maximum degree in the network. Due to sparsity of matrix W, the calculation of the value $(W^3)_{ii}$ requires $O(\Delta^3)$ multiplication operation without the use of any accelerating technique in matrix multiplication which —we believe— could further reduce the cost. So the total cost for calculating g_l is in the worst case $O(\Delta^3)$. Since we have m links, the overall complexity would be $O(\Delta^3 m T_{conv})$ where T_{conv} is the number of iterations needed for the gradient to converge (i.e., to be smaller than a given threshold). Notice that the complexity for solving the SDP for (2.6) is of order $O(m^3)$ where m is the number of links in the network. Therefore, on networks where $\Delta << m$, the gradient method could be computationally more efficient given that T_{conv} is not very large.

- Communication Complexity: Two packets are transmitted by each node on each link at steps 1 and 2. So the complexity would be two messages per link per node and per iteration. The first message carries at most Δ values (the weight vector $\mathbf{W}_i$) and the second message carries one real value ($(W^3)_{ii}$).

2.6 Performance Evaluation

In this section we evaluate the different optimization algorithms (Newton, gradient, etc.) studied in this chapter. We also evaluate the speed of convergence of consensus protocols when the weight matrix W is selected according to our proposed Schatten norm minimization. As we have discussed so far, this speed is asymptotically determined by the second largest eigenvalue in module ($\mu(W)$), that will be one of two performance metrics considered here. For the other metric, we define

the convergence time to be the number of iterations needed for the error (the distance between the estimates and the actual average) to become smaller than a given threshold. More precisely, we define the normalized error $e(k)$ as

$$e(k) = \frac{||\mathbf{x}(k) - \bar{\mathbf{x}}||_2}{||\mathbf{x}(0) - \bar{\mathbf{x}}||_2}, \tag{2.40}$$

where $\bar{\mathbf{x}} = x_{ave}\mathbf{1}$.

Additionally, we carry on simulations to study the effect of a *topological optimization* (by adding two-hops links in the graph) compared to the weight optimization on the links. The simulations are done on random graphs (Erdös-Renyi (ER) graphs and Random Geometric Graphs (RGG)) and on two real networks (the Enron company internal email exchange network [SA04] and the dolphin social network [LSB+03]). The random graphs are generated as following :

- For the ER random graphs $ER(n, Pr)$, we start from n nodes fully connected graph, and then every link is removed from the graph by a probability $1 - Pr$ and is left there with a probability Pr. We have tested the performance for different probabilities Pr.
- For the RGG random graphs, n nodes are thrown uniformly at random on a unit square area, and any two nodes within a connectivity radius r are connected by a link. We have tested the performance for different values of the connectivity radius. It is known that for a small connectivity radius, nodes tend to form clusters.

The description of the two real datasets follows:

- The 151 nodes in the Enron dataset correspond to different employees of the company and an edge in the graph refers to an exchange of emails between two employees (only internal emails within the company are considered where at least 3 emails are exchanged between two nodes in this graph).
- The dolphin social network is an undirected social network of frequent associations between 62 dolphins in a community living off Doubtful Sound, New Zealand.

2.6.1 Newton versus Gradient methods for Schatten p-Norm Minimization

We apply the optimization techniques developed in this chapter to solve problem (2.10) on Erdos Renyi random networks. We compare the number of iterations for convergence of Newton's method with those of first order methods like the Descent Gradient (DG) and the accelerated gradient method (due to Nesterov [Nes04]) using either backtracking line search (denoted by BT-methods in the figure) or exact line search (denoted by Exact-methods in the figure).[4] The Descent

[4]We implemented directly the methods in Matlab.

T_{conv} (number of iterations)	$ER(n = 100, Pr = 0.07)$			
	$p = 2$	$p = 4$	$p = 6$	$p = 10$
Exact-Newton	1	5	5.7	6.1
Pure-Newton	1	9	11.1	13.9
Exact-DG	72.3	230.5	482.7	1500.5
Exact-Nesterov	130.2	422.8	811.3	1971.2
BT-DG or BT-Nesterov	> 5000	> 5000	> 5000	> 5000

Table 2.1: Convergence time using different optimization methods for problem (2.10).

Gradient method follows the same steps of the Newton's algorithm (Section 2.4.6), but in Step 2, the Hessian H is taken as the identity matrix (for Descent Gradient methods $H_{DG} = I_m$). The accelerated gradient (Nesterov) is as follows, starting by $\mathbf{w}^{(0)} = \mathbf{w}^{(-1)} = \mathbf{0} \in \mathbb{R}^m$, the iterations are given by:

$$\mathbf{y} = \mathbf{w}^{(k-1)} + \frac{k-2}{k+1}(\mathbf{w}^{(k-1)} - \mathbf{w}^{(k-2)});$$

$$\mathbf{w}^{(k)} = \mathbf{y} - t^{(k)}\nabla_{\mathbf{y}} f(\mathbf{y}),$$

where $t^{(k)}$ is the stepsize. The Nesterov algorithm usually achieves faster rate of convergence (asymptotically) with respect to traditional first order methods. Since at the optimal value w^* the gradient vanishes (i.e., $||\mathbf{g}^{(k)}|| = 0$), we consider the convergence time T_{conv} to be:

$$T_{conv} = \min\{k : ||\mathbf{g}^{(k)}|| < 10^{-10}\}.$$

Table 2.1 shows the results for the Newton's and the other first order methods. The initial condition for the optimization is given by $W^{(0)} = I_n$ which is a feasible starting point. The values are averaged over 100 independent runs for each of the (n, Pr, p) values. The results show that the average convergence time of Newton's method is shorter than that of the first order methods in terms of the number of iterations. As we can see, when using exact line search, Exact-Nesterov is slower than Exact-DG method, this can be due to the fact that the Descent Gradient does not suffer from the zig-zag problem usually caused by poorly conditioned convex problems. Moreover, using backtracking line search for first order methods is not converging in a reasonable number of iterations because the function we are considering is not Lipschitz continuous when $p > 2$ and because of the high precision stopping condition. Note that, the number of iterations is not the only factor to take into account, in fact the Newton's method requires at each iteration to invert the Hessian matrix, while DG has lower computational cost. However, DG is very sensitive to changing the stepsize, while Newton's method is not. By applying constant or backtracking line search stepsizes to the DG method, the algorithm is not converging in a reasonable number of iterations while even the simplest Newton's method (pure Newton that uses a stepsize equals to 1 for all iterations) is converging in less than 14 iterations for the $ER(n = 100, Pr = 0.07)$ graphs.

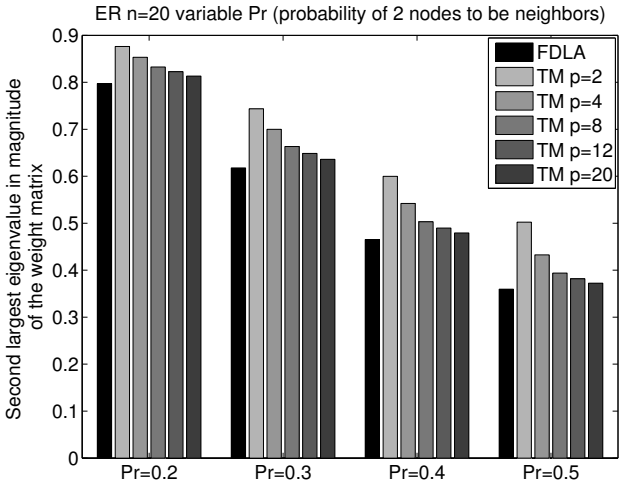

Figure 2.2: Performance comparison between the optimal solution of the FDLA problem (labeled FDLA) and the approximated solutions obtained solving the Schatten Norm minimization for different values of p (labeled TM).

2.6.2 Comparison of the Schatten Norm Solution with the Optimal Solution

We first compare $\mu\left(W_{(p)}\right)$ for the solution $W_{(p)}$ of the Schatten p-norm (or Trace) minimization problem (2.10) with its minimum value obtained solving the symmetric FDLA problem (2.6). To this purpose we used the `CVX` solver (see Section 2.2). This allows us also to evaluate how well problem (2.10) approximates problem (2.6) for finite values of the parameter p. The results in Fig. 2.2 have been averaged over 100 random graphs with 20 nodes generated according to the Erdos-Renyi (ER) model, where each link is included with probability $Pr \in \{0.2, 0.3, 0.4, 0.5\}$. We see from the results that as we solve the trace minimization for larger p, the asymptotic convergence speed of our approach converges to the optimal one as proven in Proposition 2.

2.6.3 Other Distributed Approaches: Asymptotic Convergence Rate

We compare now our algorithm for $p = 2$ and $p = 4$ with other distributed weight selection approaches described in Section 2.2.

Fig. 2.3 shows the results on connected Erdös-Renyi (ER) graphs and Random Geometric Graphs (RGG) with 100 nodes for different values respectively of the probability Pr and of the connectivity radius r. We provide 95% confidence intervals by averaging each metric over 100 different samples. We see in Fig. 2.3 that TM for $p = 2$ and $p = 4$ outperforms other weight selection algorithms on ER by giving lower μ. Similarly on RGG the TM algorithm reaches faster convergence than the other known algorithms even when the graph is well connected (large connectivity

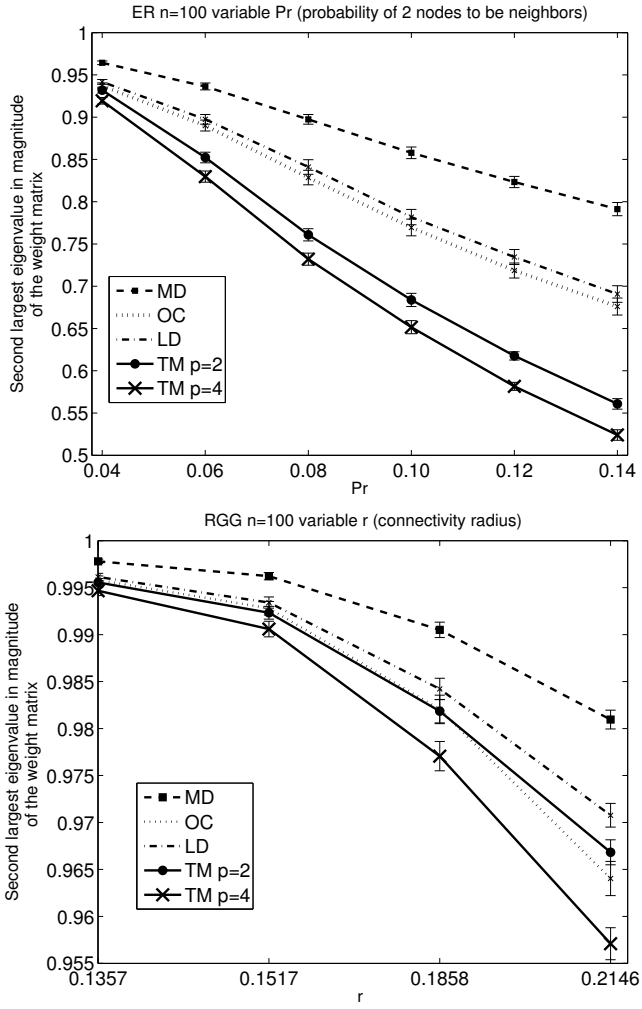

Figure 2.3: Performance comparison between Schatten Norm minimization (TM) for $p = 2$ and $p = 4$ with other weight selection algorithms on ER and RGG graphs.

radius). However, the larger the degrees of nodes, the higher the complexity of our algorithm. Interestingly even performing trace minimization for the smallest value $p = 2$ nodes are able to achieve faster speed of convergence than a centralized solution like the OC algorithm.

Apart from random networks, we performed simulations on two real world networks: the Enron company internal email exchange network [SA04] and the dolphin social network [LSB+03]. The table below compares the second largest eigenvalue μ for the different weight selection algorithms on these networks:

	MD	OC	LD	TM p=2	TM p=4
Enron μ	0.9880	0.9764	0.9862	0.9576	0.9246
Dolphin μ	0.9867	0.9749	0.9796	0.9751	0.9712

The results show that for Enron network, our totally distributed proposed algorithm TM for p=4 has the best performance ($\mu = 0.9246$) among the algorithms considered, followed by TM for p=2 ($\mu = 0.9576$) because they have the smallest μ. On the Dolphin network, again TM for p=4 has the smallest μ ($\mu = 0.9712$) but OC has the second best performance ($\mu = 0.9749$) but TM for p=2 ($\mu = 0.9751$) has similar performance to OC.

2.6.4 Communication Overhead for Local Algorithms

Until now we evaluated only the asymptotic speed of convergence, independent from the initial values $x_i(0)$, by considering the second largest eigenvalue $\mu(W)$. We want to study now the transient performance. For this reason, we consider in this subsection a random initial distribution of nodes' values and we study the performance using the convergence time metric (the number of iterations needed for the error $e(k)$ given in (2.40) to become smaller than a given threshold), i.e., the convergence time is the minimum number of iterations after which $e(k) < 0.001$ (note that $e(k)$ is non increasing).

We have shown that the weight matrix with minimum Schatten norm allows nodes to converge faster than the other heuristics, and then to exchange less messages, if a mechanism is implemented to stop consensus when estimates are close enough to the actual average. At the same time, the Schatten norm minimization algorithm may require itself a large number of messages to calculate the weights, while other local weight selection algorithms, like MD or LD, require a negligible communication exchange. In order to have a fair comparison, it is important then to consider on how many "consensus rounds" the additional communication overhead of our algorithm can be amortized.[5] Therefore, the more stable the network, the more one is ready to invest for the optimization at the beginning of consensus.

The communication overhead of the local algorithms is plotted in Fig. 2.4. For

[5]For example, the consensus round of the daily average temperature in a network of wireless environmental monitoring sensors is one day because every day a new averaging consensus algorithm should be run.

each algorithm we consider the following criteria to define its communication overhead. First we consider the number of messages that should be exchanged in the network for the weight optimization algorithm to converge. For example, in our networking settings (RGG with 100 nodes and connectivity radius 0.1517) the initialization complexity of MD algorithm is 30 messages per link because the maximum degree can be obtained by running a maximum consensus algorithm that converges after a number of iterations equal to the diameter (the average diameter for the graphs was 15 hops), while with LD the nodes only need to send their degrees to their neighbors. The communication complexity is then only 2 messages per link, the smallest among the algorithms considered. The trace minimization algorithm complexity is defined by the number of iterations needed for the gradient method to converge, multiplied by the number of messages needed per iteration as mentioned in the complexity section. In our networking setting, the TM for $p = 2$ took on average 66.22 messages per link to converge while the TM for $p = 4$ took 1388.28 messages.[6] Notice that OC depends on global values (eigenvalues of the laplacian of the graph) and is not included here because it is not a local algorithm, i.e., the weights cannot be calculated with simple iterative local methods.

In addition to the initialization complexity, we add the communication complexity for the consensus rounds. We consider that the convergence of the consensus is reached when the consensus error of Eq. (2.40) drops below 0.1%. The total communication overhead of the local algorithms is plotted in Fig. 2.4. The figure shows the total number of messages transmitted on a link, considering both those needed initially to calculate the weights and those needed to determine the average with a relative error from consensus precision (10^{-3}). The TM algorithms have high initial communication overhead (due to the slow convergence of the gradient method for weight calculation), but then the more the consensus rounds we have the more the messages are saved in comparison to the simpler methods. Note that the asymptotic results are reflected in the slopes of the lines. As the figure shows, if the network is used for more than 8 consensus rounds then TM $p = 4$ is recommended, while TM $p = 2$ starts outperforming LD and MD already for 2 consensus rounds.

2.6.5 Joint Consensus-Optimization (JCO) Procedure

In the following experiments we address also another practical concern. It may seem our approach requires to wait for the convergence of the iterative weight selection algorithm before being able to run the consensus protocol. This may be unacceptable in some applications specially if the network is dynamic and the weights need to be calculated multiple times. In reality, at each slot the output of the distributed Schatten norm minimization is a new feasible weight matrix, that can be used by the consensus protocol, and (secondarily) should also have faster convergence properties than the one at the previous step. It is then possible to interleave the weight optimization steps and the consensus averaging ones: at a given slot each node

[6] The step size γ_k is calculated with values $a = 10/p$ and $b = 100$, and convergence is obtained when $||g||$ drops below the value 0.02.

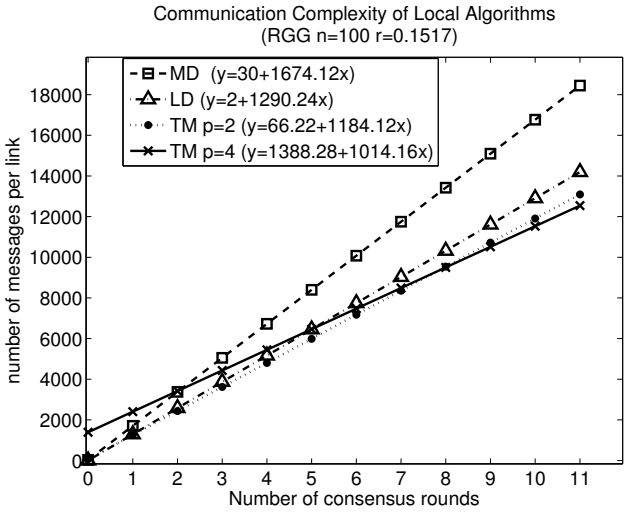

Figure 2.4: Communication overhead of local algorithms.

will improve its own weight according to (2.30) and use the current weight values to perform the averaging (2.1). We refer to this algorithm as the joint consensus–optimization (JCO) procedure. Weights can be initially set according to one of the other existing algorithms like LD or MD. The convergence time of JCO depends also on the choice of the stepsize, that is chosen to be $\gamma^{(k)} = \frac{1}{p(1+k)}$.

The simulations show that our weight selection algorithm outperforms the other algorithms also in this case. In particular, Fig. 2.5 shows the convergence time for various weight selection criteria on ER and RGG graphs. For each of the network topology selected, we averaged the data in the simulation over 100 generated graphs, and for each of these graphs we averaged the convergence time of the different algorithms over 20 random initial conditions (the initial conditions were the same for all algorithms). Notice that running at the same time the optimization with consensus gave good results in comparison to LD, MD, and even OC algorithms. We also notice, that the initial selection of the weights does not seem to have an important role for the TM-JCO approach. In fact, despite the LD weight matrix leads itself a significantly faster convergence than the MD weight matrix, initializing the TM method with the LD weight matrix or with the MD weight matrix leads only to minor differences (compare the results for TM-JCO-LD and TM-JCO-MD), suggesting that the weight optimization algorithm moves fast away from the initial condition.

2.6.6 Topology versus Weight Optimization

We turn our attention in this section to the effect of the topology on the performance. The main optimization problem (2.6) considers a fixed topology and optimizes the weights on top of this topology where simple algorithms do not provide any guaran-

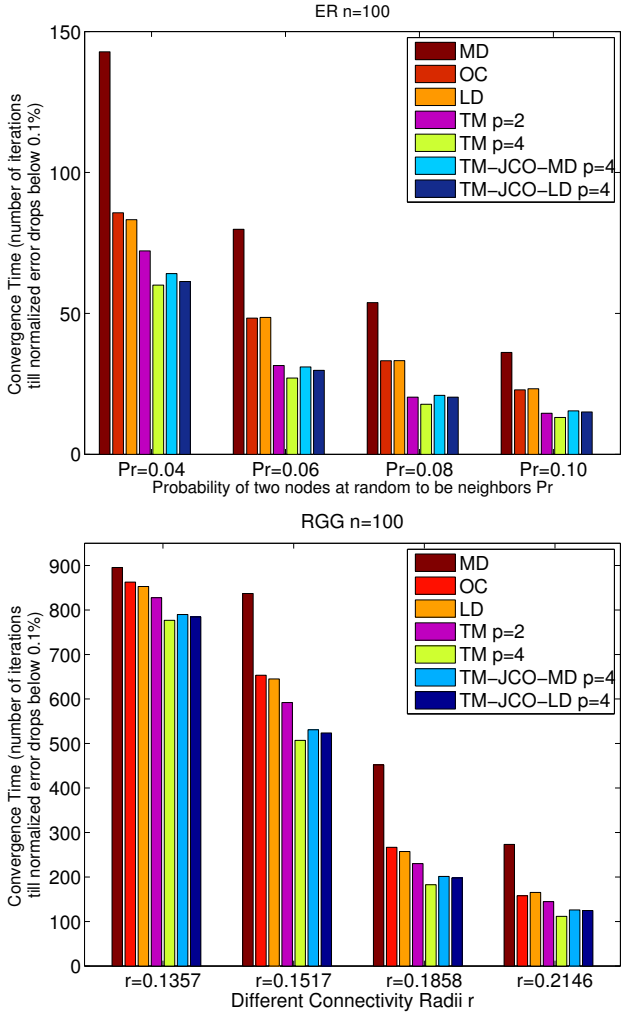

Figure 2.5: Convergence time of different weight selection algorithms on ER and RGG graphs. TM-JCO-LD $p = 4$ is the joint consensus-optimization algorithm initialized with the LD algorithm's weight matrix and the same for TM-JCO-MD $p = 4$ but initialized with the MD algorithm's one.

	RGG $n = 50$		ER $n = 50$	
	$r = 0.25$	$r = 0.3$	$Pr = 0.08$	$Pr = 0.12$
$\mu({}_{G}W_{(FDLA)})$	0.9390	0.8668	0.8511	0.7241
$\mu({}_{G^2}W_{(LD)})$	0.9070	0.8058	0.8328	0.7144

	Ring		Grid		Enron
	$n = 50$	$n = 100$	$n = 36$	$n = 64$	
$\mu({}_{G}W_{(FDLA)})$	0.9921	0.9980	0.9210	0.9210	0.8287
$\mu({}_{G^2}W_{(LD)})$	0.9843	0.9961	0.8523	0.9155	0.8208

Table 2.2: The effect of Graph Density versus Weight Optimization on the speed of convergence of consensus protocols. The table shows the comparison on different graph topologies between the speed of convergence of: **(1)** the simple weight selection algorithm (LD) on the graph G^2 quantified by $\mu({}_{G^2}W_{(LD)})$ and **(2)** the best weight selection algorithm (FDLA) on the graph G quantified by $\mu({}_{G}W_{(FDLA)})$.

tee on the speed of convergence, and more sophisticated ones are resource consuming because they select the weights solving complex optimization problems. In this part we evaluate if simple changes to the network topology may speed up the convergence of consensus protocols more than complex weight optimization techniques. In particular we compare the performance of the average consensus protocol in the two following scenarios. In the first scenario, the topology is unchanged and weights are selected according to commonly used algorithms, including those that guarantee faster convergence to consensus. In the second scenario, the simplest weights selection algorithms are used, but direct links to 2-hop away nodes are added to the original graph G, then shrinking by 2 the network diameter. We denote by G^2 (the *square graph*) this denser graph. Practically speaking this topological change does not require to really add new links: it can be obtained by forwarding nodes' local variables 2-hops away, so that a generic node i is aware of all the nodes' beliefs in the extended neighborhood $N_i^{G^2} = \cup_{j\in\{N_i,i\}} N_j$. In what follows we are going to consider this way to operate. It has also the advantage to allow us to quantify the cost of the topological change in terms of an increase of communication overhead. The comparison is carried on for different graph topologies: rings, square Grids, random graphs (Erdos-Renyi with link existence probability Pr), Random Geometric graphs (with connectivity radius r), and real world network topologies as Enron internal email exchange network [SA04]. For convenience, let A be a weight selection algorithm for the average consensus protocol,[7] we denote by ${}_{G}W_{(A)}$ the weight matrix generated by A on the graph G. In order to evaluate the effect of weight selection algorithms and of topology on convergence speed, we are going to compare $\mu({}_{G}W_{(A)})$, where A is the optimal weight selection algorithm (that solves problem (2.6)) or one of its approximations, and $\mu\left({}_{G^2}W_{(B)}\right)$, where B is a simpler weight selection algorithm.

[7] A can be any one of the following weight selection algorithms presented earlier: MD, LD, OC, TM $p = 2$, or FDLA.

We first compare the performance (the asymptotic speed of convergence) of the consensus protocol on the denser graph G^2 when weights are selected according to the LD algorithm with the performance on the original graph G when the optimal weight selection algorithm FDLA is used. Results in Table 2.2 show that on all the topologies considered $\mu({}_{G^2}W_{(LD)}) < \mu({}_{G}W_{(FDLA)})$ and then $\mu({}_{G^2}W_{(LD)}) < \mu({}_{G}W_{(A)})$ for any algorithm A. Then the higher graph density provides a more significant improvement than the optimal choice of links weights.

We now evaluate the communication overhead of the two approaches in terms of the number of messages sent. Equation (2.1) requires that nodes at each iteration k to use the local variables of their neighbors (node i uses $x_j(k)$ for all $j \in N_i^G$). Therefore, each node must receive at every iteration these values and the total number of messages M sent in the system will be $M = 2 \times m$ where m is the number of links in the graph. On G^2, N_i in equation (2.1) is replaced by $N_i^{G^2}$. As we mentioned above, it is possible to mimic the consensus protocol on G^2 using only the links in G. In this case the operation requires 2 steps. First each node broadcasts its belief to its neighbors in N_i. Then, each node sends another broadcast message to its neighbors in N_i with all the beliefs that it has collected during the first step. In this way every node gets to know the beliefs of all the nodes in $N_i^{G^2}$. The total number of messages is then twice as larger than in the first scenario.[8] For this reason, we decided to compare the speed of convergence in the two scenarios when the number of messages being equal. This corresponds to consider that the consensus protocol on G performs two weighted linear iterations according to (2.1) for each linear iteration on G^2. Another possible way to interpret this comparison is that if the duration of an iteration is determined by the time needed to transmit one message on a link, then a consensus protocol iteration on G^2 requires twice as much time than one on G. It is easy to evaluate the speed of convergence of the "accelerated" consensus protocol that performs two linear iterations every time unit. In fact it can be checked that this corresponds to use as weight matrix $({}_{G}W_{(A)})^2$ [BGPS06]. Then, the asymptotic speed of convergence is determined by $\mu({}_{G}W_{(A)}^2)$. Note that the following equation holds: $\mu({}_{G}W_{(A)}^2) = \left(\mu({}_{G}W_{(A)})\right)^2$. Simulation results in Table 2.3 show that $\mu({}_{G^2}W_{(A)}) < \mu({}_{G}W_{(A)}^2)$ for any algorithm A introduced in the previous sections. Then the denser topology leads to faster convergence speed even when the number of messages is equivalent. For this reasons, simple weight selection algorithms as LD on G^2 can still outperform more complex ones like $TM - 2$ or OC (the results from Table 2.3 show that $\mu({}_{G^2}W_{(LD)}) < \mu({}_{G}W_{(TM-2)}^2)$ and $\mu({}_{G^2}W_{(LD)}) < \mu({}_{G}W_{(OC)}^2)$ on most of the topologies) and also achieve in some cases results very similar to FDLA (e.g., on the grid).

The nutshell of the simulations on these graphs is given by two main interesting results. The first result is that simple weight selection algorithms can achieve sig-

[8] We observe here that the messages sent in the second step have usually a larger data payload than those sent in the first step, because they carry many belief values. Here we assume that the number of messages is an adequate metric to evaluate the performance, as for example is the case if the packet header is much larger than the data payload for this application.

	RGG $n = 50$		ER $n = 50$	
	$r = 0.25$	$r = 0.3$	$Pr = 0.08$	$Pr = 0.12$
$\mu({}_GW^2_{(MD)})$	0.9665	0.9274	0.9036	0.8327
$\mu({}_{G^2}W_{(MD)})$	0.9319	0.8577	0.8967	0.7923
$\mu({}_GW^2_{(LD)})$	0.9493	0.8951	0.8591	0.7572
$\mu({}_{G^2}W_{(LD)})$	0.9070	0.8058	0.8328	0.7144
$\mu({}_GW^2_{(OC)})$	0.9378	0.8677	0.8363	0.7276
$\mu({}_{G^2}W_{(OC)})$	0.8761	0.7543	0.8177	0.6650
$\mu({}_GW^2_{(TM-2)})$	0.9419	0.8800	0.8334	0.6749
$\mu({}_{G^2}W_{(TM-2)})$	0.8900	0.7565	0.7078	0.4590
$\mu({}_GW^2_{(FDLA)})$	0.8817	0.7513	0.7244	0.5243
$\mu({}_{G^2}W_{(FDLA)})$	0.7591	0.5478	0.5219	0.3098

	Ring		Grid		Enron
	$n = 50$	$n = 100$	$n = 36$	$n = 64$	
$\mu({}_GW^2_{(MD)})$	0.9894	0.9974	0.8957	0.9401	0.9761
$\mu({}_{G^2}W_{(MD)})$	0.9843	0.9961	0.8730	0.9240	0.9057
$\mu({}_GW^2_{(LD)})$	0.9894	0.9974	0.8876	0.9364	0.9726
$\mu({}_{G^2}W_{(LD)})$	0.9843	0.9961	0.8523	0.9155	0.8208
$\mu({}_GW^2_{(OC)})$	0.9843	0.9960	0.8662	0.9239	0.9534
$\mu({}_{G^2}W_{(OC)})$	0.9751	0.9937	0.7919	0.8776	0.8277
$\mu({}_GW^2_{(TM-2)})$	0.9894	0.9974	0.8857	0.9359	0.9143
$\mu({}_{G^2}W_{(TM-2)})$	0.9843	0.9961	0.8403	0.9119	0.5568
$\mu({}_GW^2_{(FDLA)})$	0.9843	0.9960	0.8482	0.9126	0.6868
$\mu({}_{G^2}W_{(FDLA)})$	0.9691	0.9922	0.7241	0.8343	-

Table 2.3: The 2-hop averaging topology optimization in G can be done by sending 2-hop messages. Every averaging iteration in this case (having speed governed by $\mu({}_{G^2}W_{(A)})$) consumes as many messages as two iterations of normal averaging on G (the speed of two iteration averaging is governed by $\mu({}_GW^2_{(A)})$). Since $\mu({}_{G^2}W_{(A)}) < \mu({}_GW^2_{(A)})$ for any weight selection algorithm A and any network, the 2-hop averaging can have a significant faster convergence speed than standard averaging while sending the same number of messages.

nificantly faster convergence on the denser graph G^2 than any weight optimization technique on the original graph G. This improvement comes at the cost of an increase of communication overhead in the network. Our second (less expected) result is that, for a given weight selection algorithm, the convergence is faster on G^2 than on G even when the number of messages is equal. Because of this, simpler weight selection algorithms on G^2 can achieve performance similar to more complex ones on G. These results suggest that topological optimization can have a more important role than weight optimization techniques to speed up information propagation.

2.7 Stability and Misbehaving Nodes

In this section we first explain how the convergence of the consensus protocol can be guaranteed also for "small" p values (see the remark in section 2.3) and then we discuss how to deal with some forms of nodes' misbehavior.

2.7.1 Guaranteeing Convergence of Trace Minimization

The conditions (2.3)-(2.5) guarantee that the consensus protocol converges to the correct average independently from the initial estimates. In this section, for the sake of conciseness, we call a weight matrix that satisfies these set of conditions a *convergent matrix*. A convergent matrix is then any matrix that guarantees the convergence of average consensus protocols. We showed in Proposition 2 that for p large enough, the solution $W_{(p)}$ of (2.10) is a convergent matrix. However, for "small" p values, it may happen that $\mu(W_{(p)}) \geq 1$ (the other conditions are intrinsically satisfied) and then the consensus protocol does not converge for all the possible initial conditions. We observe that if all the link weights and the self weights in $W_{(p)}$ are strictly positive then $W_{(p)}$ is a convergent matrix. In fact from Perron-Frobenius theorem for nonnegative matrices [Sen06] it follows that a stochastic weight matrix W for a strongly connected graph where $w_{ij} > 0$ if and only if $(i,j) \in E$ satisfies (2.5) (i.e., $\mu(W) < 1$). Then, the matrix may not be convergent only if one of the weights is negative. Still in such a case nodes can calculate in a distributed way a convergent weight matrix that is "close" to the matrix $W_{(p)}$. In this section we show how it is possible and then we discuss a practical approach to guarantee convergence while not sacrificing the speed of convergence of $W_{(p)}$ (when it converges).

We obtain a convergent matrix from $W_{(p)}$ in two steps. First, we project $W_{(p)}$ on a suitable set of matrices that satisfy conditions (2.3) and (2.5), but not necessarily (2.4), then we generate a symmetric convergent matrix from the projection. Let $\hat{W} = W_{(p)}$ be the matrix to project, the solution of the following projection is guaranteed to satisfy (2.3) and (2.5):

$$\begin{aligned} \underset{W}{\text{argmin}} \quad & ||W - \hat{W}||_F^2 \\ \text{subject to} \quad & W\mathbf{1} = \mathbf{1}, \\ & W \in \mathcal{C}'_G, \end{aligned} \tag{2.41}$$

where $\mathcal{C}'_G$ is the set of non-negative matrices such that $w_{ij} \geq \delta > 0$ if $(i,j) \in E$, $w_{ij} = 0$ if $(i,j) \notin E$, and $||.||_F$ is the Frobenius matrix norm. The constant $\delta > 0$ is a parameter that is required to guarantee that the feasible set is closed.

Now, we show how it is possible to project a matrix $\hat{W}$ according to (2.41) in a distributed way. We observe that this approach is feasible because we do not require the projected matrix to be symmetric (and then satisfy (2.4)). The key element for the distributed projection is that the Frobenius norm is separable in terms of the variables $\mathbf{W}_i$ (the $d_i \times 1$ vector of weights selected by node i for its neighbors), so that problem (2.41) is equivalent to:

$$\begin{aligned} \underset{\mathbf{W}_1,\ldots,\mathbf{W}_n}{\text{argmin}} \quad & \sum_{i=1}^{n} r(\mathbf{W}_i) \\ \text{subject to} \quad & \mathbf{W}_i^T \mathbf{1}_{d_i} \leq 1 \quad \forall i, \\ & \mathbf{W}_i \geq \delta > 0 \quad \forall i, \end{aligned} \tag{2.42}$$

where $\mathbf{1}_{d_i}$ is the $d_i \times 1$ vector of all ones, and $r(\mathbf{W}_i)$ is defined as follows:

$$r(\mathbf{W}_i) = (w_{ii} - \hat{w}_{ii})^2 + \sum_{j \in N_i} (w_{ij} - \hat{w}_{ij})^2 \tag{2.43}$$

$$= (\mathbf{W}_i - \hat{\mathbf{W}}_i)^T (\mathbf{W}_i - \hat{\mathbf{W}}_i) + \left((\mathbf{W}_i - \hat{\mathbf{W}}_i)^T \mathbf{1}_{d_i} \right)^2 \tag{2.44}$$

$$= (\mathbf{W}_i - \hat{\mathbf{W}}_i)^T \left(I_{d_i} + \mathbf{1}_{d_i} \mathbf{1}_{d_i}^T \right) (\mathbf{W}_i - \hat{\mathbf{W}}_i), \tag{2.45}$$

where I_{d_i} is d_i-identity matrix. Since the variables in (2.42) are separable in $\mathbf{W}_1, \ldots, \mathbf{W}_n$, then each node i can find the global solution for its projected vector $\mathbf{W}_i^{(proj)}$ by locally minimizing the function $r(\mathbf{W}_i)$ subject to its constraints.

Once the weight vectors $\mathbf{W}_i^{(proj)}$ are obtained, the projection of $W_{(p)}$ on the set $\mathcal{C}'_G$ is uniquely identified. We denote it $W^{(proj)}$. We can then obtain a convergent weight matrix $W^{(conv)}$ by modifying $W^{(proj)}$ as follows. For every link $l \sim (i,j)$, we set:

$$w_l^{(conv)} = \min \left\{ \left(\mathbf{W}_i^{(proj)} \right)_{\alpha(j)}, \left(\mathbf{W}_j^{(proj)} \right)_{\alpha(i)} \right\},$$

where $\alpha(j)$ (similarly $\alpha(i)$) is the index of the node j (similarly i) in the corresponding vector. Then we calculate the convergent weight matrix:

$$W^{(conv)} = I - Q \times \text{diag}(\mathbf{w}^{(\mathbf{conv})}) \times Q^T.$$

While the matrix $W^{(conv)}$ is convergent, its speed of convergence may be slower than the matrix $W_{(p)}$, assuming this converges too. Then the algorithm described above should be ideally limited to the cases where $W_{(p)}$ is known to not be convergent. Unfortunately in many network scenarios this may not be known a priori. We discuss a possible practical approach in such cases. Nodes first compute $W_{(p)}$. If all the link-weights and self-weights are positive then the matrix $W_{(p)}$ can be used in the consensus protocol without any risk. If one node has calculated a non-positive

weight, then it can invoke the procedure described above to calculate $W^{(conv)}$. Nodes can then run the consensus protocol using only the matrix $W^{(conv)}$ at the price of a slower convergence or they can run the two consensus protocols in parallel averaging the initial values both with $W^{(conv)}$ and $W_{(p)}$. If the estimates obtained using $W_{(p)}$ appear to be converging to the same value of the estimates obtained using $W^{(conv)}$, then the matrix $W_{(p)}$ is likely to be convergent and the corresponding estimates should be closer to the actual average.[9]

2.7.2 Networks with Misbehaving Nodes

The convergence of the average consensus relies on all the nodes correctly performing the algorithm. If one node transmits an incorrect value, the estimates of all the nodes can be affected. In this section we address this particular misbehavior. In particular, let $x_i(k)$ be the estimate of node i at iteration k, if $x_i(k) \neq w_{ii}(k-1)x_i(k-1) + \sum_{j \in N_i} w_{ij}(k-1)x_j(k-1)$, then we call i a *misbehaving node*. *Stubborn* nodes are a special class of misbehaving nodes that keep sending the same estimate at every iteration (i.e., a node i is a stubborn node when at every iteration k we have $x_i(k) = x_i(k-1) \neq w_{ii}(k-1)x_i(k-1) + \sum_{j \in N_i} w_{ij}(k-1)x_j(k-1)$). The authors of [ACFO11] and [BABJ12] showed that networks with stubborn nodes fail to converge to consensus. In [BABJ12], they proposed a robust average consensus algorithm that can be applied on networks having one stubborn node and converges to consensus. To the best of our knowledge, dealing with multiple stubborn nodes is still an open issue. It turns out that with a minor modification of our JCO algorithm, the nodes can detect an unbounded number of misbehaving nodes under the following assumptions:

- **Assumption 1:** There is no collusion between misbehaving nodes (every node, even a misbehaving one, that detects a misbehaving neighbor declares it).
- **Assumption 2:** At each iteration a misbehaving node sends the same (potentially wrong) estimate to all its neighbors.

The second assumption can be automatically satisfied in the case of a broadcast medium.

In the JCO procedure in section 2.6.5, nodes perform one weight optimization step and one average consensus step at every iteration. Consider an iteration k, weight optimization requires nodes to receive the weight vectors used by their neighbors (in particular, node i will receive $\mathbf{W}_j^{(k-1)}$ from every neighbor $j \in N_i$), and the averaging protocol requires them to receive their neighbors estimates (in particular, node i will receive $x_j(k)$ from every neighbor $j \in N_i$). We also require that nodes send the estimates of their neighbors, e.g., node i will receive together with the vector $\mathbf{W}_j^{(k-1)}$ another vector $\mathbf{X}_j(k-1)$ from every neighbor $j \in N_i$ where $\mathbf{X}_j(k-1)$

[9] Note that if $\mu(W_{(p)}) > 1$ the estimates calculated using $W_{(p)}$ diverge in general, then it should be easy to detect that the two consensus protocols are not converging to the same value.

is the vector of the estimates of the neighbors of node j. With such additional information, the following simple algorithm allows nodes to detect a misbehaving neighbor:

Misbehaving Neighbor Detection Algorithm - Node i

$\{x_j(k), \mathbf{X}_j(k-1), \mathbf{W}_j^{(k-1)}\}$: the message received from a neighbor j at iteration k

$\alpha(i)$: index of a node i in the corresponding vector

for all $j \in N_i$

$C = w_{jj}(k-1)x_j(k-1) + \mathbf{X}_j^T(k-1)\mathbf{W}_j^{(k-1)}$

if $(x_j(k) \neq C)$ or $\left(x_i(k-1) \neq (\mathbf{X}_j(k-1))_{\alpha(i)}\right)$

or $\left(w_{ij}(k-1) \neq \left(\mathbf{W}_j^{(k-1)}\right)_{\alpha(i)}\right)$

Declare j as misbehaving node.

end if

end for

The first condition $(x_j(k) \neq w_{jj}(k-1)x_j(k-1)+\mathbf{X}_j^T(k-1)\mathbf{W}_j^{(k-1)})$ corresponds to the definition of a misbehaving node and allows neighbors to detect a node sending a wrong estimate. The second and third conditions $(x_i(k-1) \neq (\mathbf{X}_j(k-1))_{\alpha(i)})$ or $(w_{ij}(k-1) \neq \left(\mathbf{W}_j^{(k-1)}\right)_{\alpha(i)})$ detect if node j is modifying the content of any element in the vectors $\mathbf{X}_j(k-1)$ and $\mathbf{W}_j^{(k-1)}$ before sending them to its neighbors. More precisely, because of Assumption 2, if a node changes any element in the previously mentioned vectors, then this message will reach all neighbors including the neighbors concerned by this modification. These neighbors will remark this modification by checking the second and the third conditions, and, due to Assumption 1, they will declare the node as misbehaving.

Once a node is declared a misbehaving node, the others can ignore it by simply assigning a null weight to its links in the following iterations.

2.8 More on Schatten p-Norm and its Relation to Machine Learning

The Schatten p-norm is often considered in machine learning for the regularization problem in applications such as multi-task learning [AMPY07], collaborative filtering [SRJ05] and multi-class classification [AFSU07] because it has some favorable properties (being orthogonally invariant for example). Up to our knowledge, an exact line search Newton's method has not yet been proposed for constrained Schatten p-norm problems in machine learning but they are usually solved by first order gradient methods. In this section, we develop Newton's method for the general norm optimization problem and we show that the weight selection by Schatten p-norm proposed in this chapter can be considered a special case of this section's more

general problem. The optimization problem we are interested in is the following:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & ||X||_{\sigma p} \\ \text{subject to} \quad & \phi(X) = \mathbf{y}, \\ & X \in \mathbb{R}^{n_1,n_2} \ , \ \mathbf{y} \in \mathbb{R}^c, \end{aligned} \tag{2.46}$$

where $||X||_{\sigma p}$ is the Schatten p-norm of the matrix X which is the L-p norm of its singular values, i.e., $||X||_{\sigma p} = (\sum_i \sigma_i^p)^{1/p}$, and $\phi(X)$ is a linear function of the elements of X. For $p = 1$, the norm is known as the nuclear norm, while for $p = \infty$ it is the spectral norm; for both values of p, problem (2.46) can be formulated as a semi-definite programming and solved using standard interior-point methods [FHB01, XB04]. The authors in [AMP10] refer to problem (2.46) as the *minimal norm interpolation* problem.

In this section, we show that for an even integer p in problem (2.46), we can easily calculate explicitly both the gradient and the Hessian by exploiting the special structure of the objective function, constraints linearity, and by carefully rewriting the Schatten norm problem by stacking the columns of the matrix to form a long vector. While we still need to invert the Hessian numerically, this matrix has lower dimension than the typical KKT matrix used in Newton's methods for solving such constrained problems.

We use the same notation for the Hessian and gradient presented in Section 2.4, i.e., for the scalar function of a *vector*, $f : \mathbb{R}^m \to \mathbb{R}$, the gradient of the function $f(\mathbf{x})$ with respect to the vector $\mathbf{x} \in \mathbb{R}^m$ is denoted by $\nabla_{\mathbf{x}} f \in \mathbb{R}^m$ and its Hessian is denoted by the matrix $\nabla^2_{\mathbf{x}} f \in \mathbb{R}^{m,m}$ whose elements are given by the following equations:

$$(\nabla_{\mathbf{x}} f)_l \triangleq \frac{\partial f}{\partial x_l}, \text{ and } \left(\nabla^2_{\mathbf{x}} f\right)_{l,k} \triangleq \frac{\partial^2 f}{\partial x_l \partial x_k}.$$

For a scalar function of a *matrix*, $h : \mathbb{R}^{n_1,n_2} \to \mathbb{R}$, the gradient of the function $h(X)$ with respect to the vector $\text{vect}(X) \in \mathbb{R}^{n_1 n_2,1}$ is denoted by $\nabla_X h \in \mathbb{R}^{n_1 n_2,1}$ and its Hessian is denoted by the matrix $\nabla^2_X h \in \mathbb{R}^{n_1 n_2, n_1 n_2}$ whose elements are given by the equations:

$$\nabla_X h_{(ij)} \triangleq \frac{\partial h}{\partial x_{ij}}, \text{ and } \nabla^2_X h_{(ij)(st)} \triangleq \frac{\partial^2 h}{\partial x_{ij} \partial x_{st}}.$$

As $\phi(X)$ in (2.46) is a linear function of the elements of X, then it can be written also as:

$$\phi(X) = A \ \text{vect}(X),$$

where $A \in \mathbb{R}^{c,n_1 n_2}$ and c is the number of constraints. We suppose that the problem admits always a solution X^*. Since we are interested in applying Newton's method to solve equation (2.46), the objective function should be twice differentiable. Not all the norms satisfy this property, we limit then our study to the case where p is an even integer because in this case we show that the problem (2.46) is equivalent to a smooth optimization problem. Let $p = 2q$, raising the objective function to

the power p will not change the solution set, so we can equivalently consider the objective function:

$$h(X) = ||X||_{\sigma p}^{p} = \mathrm{Tr}\left(\left(XX^T\right)^q\right).$$

Since we only have linear constraints ($A\ \mathrm{vect}(X) = \mathbf{y}$), by taking only the linearly independent equations, and using Gaussian elimination to have a full row rank matrix, we can rewrite the constraints as follows:

$$\left[\begin{array}{cc} I_r & B \end{array}\right] P\ \mathrm{vect}(X) = \hat{\mathbf{y}},$$

where I_r is the r-identity matrix, r is the rank of the matrix A (the number of linearly independent equations), $B \in \mathbb{R}^{r,n_1n_2-r}$, P is an $n_1n_2 \times n_1n_2$ permutation matrix of the variables, and $\hat{\mathbf{y}} \in \mathbb{R}^r$ is a vector. We arrive at the conclusion that the original problem (2.46) is equivalent to:

$$\begin{array}{ll} \underset{X}{\text{minimize}} & h(X) = \mathrm{Tr}\left(\left(XX^T\right)^q\right) \\ \text{subject to} & \left[\begin{array}{cc} I_r & B \end{array}\right] P\ \mathrm{vect}(X) = \hat{\mathbf{y}}. \end{array} \tag{2.47}$$

Before applying Newton's method to (2.47), we can further reduce the problem to an unconstrained minimization problem. By considering the equality constraints, we can form a mapping from $X \in \mathbb{R}^{n_1,n_2}$ to the vector $\mathbf{x} \in \mathbb{R}^{n_1n_2-r}$ as follows:

$$\mathbf{x} = \left[\begin{array}{cc} 0^{n_1n_2-r,r} & I_{n_1n_2-r} \end{array}\right] P\ \mathrm{vect}(X), \tag{2.48}$$

and X can be obtained from $\mathbf{x}$ and $\hat{\mathbf{y}}$ as

$$X = \mathrm{vect}^{-1}\left(P^{-1}\left[\begin{array}{c} \hat{\mathbf{y}} - B\mathbf{x} \\ \mathbf{x} \end{array}\right]\right), \tag{2.49}$$

where $\mathrm{vect}^{-1} : \mathbb{R}^{n_1n_2} \to \mathbb{R}^{n_1,n_2}$ is the inverse function of vect(), i.e., $\mathrm{vect}^{-1}(\mathrm{vect}(X)) = X$. The unconstrained minimization problem is then:

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}), \tag{2.50}$$

where $f(\mathbf{x}) = \mathrm{Tr}\left(\left(XX^T\right)^q\right)$ and X is by (2.49).

All three problems (2.46), (2.47), and (2.50) are convex and are equivalent to each other. We apply Newton's method to (2.50) to find the optimal vector $\mathbf{x}^*$ and then deduce the solution of the original problem X^*. The main difficulty in most Newton's methods is the calculation of the gradient and the Hessian. In many applications, the Hessian is not known and for this reason gradient methods are applied rather than the faster Newton's methods. However, also in this case, we show that by exploring the special structure of the function $h(X)$, we can calculate explicitly both $\nabla_{\mathbf{x}} f$ and $\nabla_{\mathbf{x}}^2 f$. To this purpose, we first calculate the gradient and Hessian of $h(X)$, and then use the linearity of the constraints. Using matrix calculus [Ber05, ORG12], and similarly to the derivation of equation (2.24), closed form expressions for the gradient and Hessian of $h(X)$ are given by the following Lemma:

Lemma 4. *Let $h(X) = Tr\left(\left(XX^T\right)^q\right)$ where $X \in \mathbb{R}^{n_1,n_2}$, then the gradient of h is given by,*

$$\nabla_X h_{(ij)} = 2q\left(\left(XX^T\right)^{q-1} X\right)_{i,j}, \tag{2.51}$$

and the Hessian,

$$\begin{aligned}\nabla^2_X h_{(ij)(st)} = 2q\sum_{k=0}^{q-2}\left(\left(XX^T\right)^k X\right)_{i,t}\left(\left(XX^T\right)^{q-2-k} X\right)_{s,j} \\ + 2q\sum_{k=0}^{q-1}\left(\left(XX^T\right)^k\right)_{i,s}\left(\left(X^TX\right)^{q-1-k}\right)_{t,j}.\end{aligned} \tag{2.52}$$

We can now apply the chain rule to calculate the gradient and Hessian of $f(\mathbf{x})$, taking into account the mapping from $\mathbf{x}$ to X in (2.49).

For the gradient $\nabla_{\mathbf{x}} f$, it holds for $l = 1, \ldots, n_1 n_2 - r$:

$$(\nabla_{\mathbf{x}} f)_l = \frac{\partial f}{\partial x_l} = \sum_{i,j} \nabla_X h_{(ij)} \frac{\partial x_{ij}}{\partial x_l}, \tag{2.53}$$

where all the partial derivatives $\frac{\partial x_{ij}}{\partial x_l}$ are constant values because (2.49) is a linear transformation. Applying the chain rule for the Hessian and considering directly that all the second order derivatives like $\frac{\partial^2 x_{ij}}{\partial x_l \partial x_k}$ are null (again because the mapping (2.49) is a linear transformation), we obtain that for $l, k = 1, \ldots, n_1 n_2 - r$:

$$\left(\nabla^2_{\mathbf{x}} f\right)_{l,k} = \frac{\partial^2 f}{\partial x_l \partial x_k} = \sum_{i,j,s,t} \nabla^2_X h_{(ij)(st)} \frac{\partial x_{ij}}{\partial x_l} \frac{\partial x_{st}}{\partial x_k}. \tag{2.54}$$

Since $f(\mathbf{x})$ is a convex function, then the calculated matrix $\nabla^2_{\mathbf{x}} f$ is semi-definite positive. We can add to the diagonals a small positive value γ to guarantee the existence of the inverse without affecting the convergence. The calculated Hessian is a square matrix having dimensions d by d where $d = n_1 n_2 - r$ may be large for some applications, and at every iteration of the Newton's method, we need to calculate the inverse of the Hessian. Efficient algorithms for inverting large matrices are largely discussed in the literature (see [IK94] for example) and are not detailed here. Nevertheless, the given matrix has lower dimension than the typical KKT matrix[10] used in Newton's method [BV04]:

$$\begin{bmatrix} \nabla^2_X h & A^T \\ A & 0 \end{bmatrix}, \tag{2.55}$$

where A is considered here to be a full row rank matrix, so the KKT matrix is a square matrix of dimensions d_{KKT} by d_{KKT} where $d_{KKT} = n_1 n_2 + r$.

Once we know the gradient $\nabla_{\mathbf{x}} f$ and the Hessian $\nabla^2_{\mathbf{x}} f$, we just apply the Newton's method to find the solution $\mathbf{x}^*$ and then obtain the solution of the original

[10] Note that the sparsity of the matrix to invert is preserved by the proposed method, i.e., if the KKT matrix is sparse due to the sparsity of A and $\nabla^2_X h$, then $\nabla^2_{\mathbf{x}} f$ is also sparse.

problem X^*. In fact, the weight selection optimization (2.10) proposed in this chapter is just a special case of the problem discussed in this section. Due to the constraint that the matrix is symmetric in (2.10), we can write the objective function as $h(W) = \mathrm{Tr}\left((WW^T)^q\right)$. Moreover, we can see that all constraints are linear equalities. Therefore, the technique derived here applies to the more specific case.

2.9 Conclusion

We have proposed in this chapter an approximated solution for the Fastest Distributed Linear Averaging (FDLA) problem by minimizing the Schatten p-norm of the weight matrix. Our approximated algorithm converges to the solution of the FDLA problem as p approaches ∞, and in comparison to it has the advantage to be suitable for a distributed implementation. We gave first a centralized implementation using Newton's method, and then we gave a totally distributed projection sub-gradient algorithm for our proposed problem. Moreover, extensive simulations on random and real networks show that the algorithm outperforms other common distributed algorithms for weight selection. We also addressed the issue of topological optimization and we compared that to the weight optimization. Finally, the issue of stubborn nodes was discussed where an appropriate algorithm to counter these malicious behaviors was proposed. We concluded this chapter by extending our approach for Schatten norm minimization to more general problems which can be of interest for machine learning applications.

Chapter 3

Consensus in the Presence of an Adversary

Contents

As we have seen so far, in consensus algorithms, nodes execute update rules to reach consensus based on neighbor to neighbor weighted average linear iterations. As in any protocol, some parameters (e.g., the weights) can be tuned faster convergence. For instance, [XB04] formulates a semi-definite program (SDP) for a *fixed* weight selection algorithm to achieve fast convergence of consensus protocols independent of initial nodes' values, and we proposed a distributed implementation for an approximation of the SDP in Chapter 2. Another approach is to design *time-varying* weights, for example [KG09, HJOV14] study finite-time consensus by arbitrary time-varying weights chosen at the time of design using matrix factorization techniques. Reference [SM12] considers dynamic weights for least mean square design in correlated or uncorrelated initial node values.

Further, networks can be susceptible to attacks from adversaries willing to drive the system away from consensus. There are different types of adversaries that can harm the network. For example compromised strategic nodes (like faulty nodes or stubborn ones [ACFO11, BABJ12]) can harm the state of the network. Other

types of strategic intervention include adversaries that cut communication links or insert noise signals in the agents' interaction protocol [KTB13]. Yet another type of adversaries inject false data (collected by nodes) into the system, which bypass bad-data detection mechanisms. False data injections are known as stealth attacks and are widely studied for the security of state estimation in electric power networks [VD14, LNR09]. In order to mitigate the effect of an adversary, security procedures should be taken into account in the design of optimal strategies in consensus protocols.

Our present work in this chapter shares with this set of references the same objectives of designing time-varying weights for faster consensus and studying optimal strategies for networks that are vulnerable to attacks. In the first part we study time-varying weights for consensus protocols within the framework of an optimal control formulation. We apply optimization techniques to obtain a locally (and possibly globally) optimizing feasible control path and provide necessary and sufficient conditions for the existence of a control that makes the system reach consensus in only one iteration. The difference with previous related work is that in this chapter we consider the initial values in our *dynamic* weight design. In the second part we study adversaries that can compromise these weights. We propose a game theoretical framework for an adversary that can add noise to the weights to drive the system away from consensus. We derive the optimal strategies using a saddle-point equilibrium (SPE) solution in mixed strategies for both players (the adversary and the network designer) in the resulting game.

The contribution of the chapter is as follows:

- We formulate, using optimal control, the problem of finding optimal weights for discrete time consensus given the information on initial conditions, and provide necessary conditions using the maximum principle for optimal design.
- Using gradient methods, we solve the weight optimization problem and provide a locally (and possibly globally) optimizing solution. We also give sufficient conditions for an optimal control to drive the system to consensus in only one stage (one iteration consensus).
- We give a game theoretical approach to model an adversary that can perturb the weights in the network. We provide the optimal strategies given by the saddle-point equilibrium of the network in mixed strategies.

3.1 Problem Formulation

In this chapter, we turn our attention to time-varying weights in consensus protocols. The system equation is then given in matrix form as follows,

$$\mathbf{x}(k+1) = W(k)\mathbf{x}(k), \tag{3.1}$$

where $W(k)$ is the weight matrix at iteration k.

Under some conditions on the weights $W(k)$, the values at the nodes are guaranteed to converge asymptotically to the average

$$\lim_{k\to\infty} \mathbf{x}(k) = \bar{\mathbf{x}},$$

where $\bar{\mathbf{x}} = x_{ave}\mathbf{1}$ and $x_{ave} = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$. We have seen that one such set of conditions is given in [XB04] with fixed weights (i.e., $W(k) = W\ \forall k$):

$$\mathbf{1}^T W = \mathbf{1}^T, \quad W\mathbf{1} = \mathbf{1}, \quad \rho(W - \frac{1}{n}\mathbf{1}\mathbf{1}^T) < 1,$$

where $\mathbf{1}$ is the vector of all ones, and $\rho(.)$ is the largest eigenvalue in magnitude of a matrix. By the first condition, the average in the network is conserved, namely

$$\mathbf{1}^T\mathbf{x}(k) = \mathbf{1}^T\mathbf{x}(0) = nx_{ave}\ \forall k, \tag{3.2}$$

the second ensures stability, and the last condition guarantees contraction on the weight matrix. At any iteration k, we define the squared error L_k from consensus as follows:

$$\begin{aligned} L_k &= \sum_{i\in V}(x_i(k) - x_{ave})^2 \\ &= (\mathbf{x}(k) - \bar{\mathbf{x}})^T(\mathbf{x}(k) - \bar{\mathbf{x}}), \\ &= \mathbf{y}_k^T\mathbf{y}_k, \end{aligned} \tag{3.3}$$

where $\mathbf{y}_k = \mathbf{x}(k) - \bar{\mathbf{x}}$.

In this chapter, we design time-varying weight matrices $W(k)$ such that consensus forms in the least number of iterations (achieving faster convergence) under the criterion of minimum squared error. Our work differs from the earlier work in the literature in that we design the weights depending on the initial values, i.e.,

$$W(k) = W(k, \mathbf{x}(0)).$$

3.2 Optimal Weight Selection on Undirected Graphs

Toward the goal stated above, since we are dealing with an undirected graph, we consider the following properties for the weight matrix for all k:

$$W(k) = W(k)^T \quad \text{and} \quad W(k)\mathbf{1} = \mathbf{1}. \tag{3.4}$$

Therefore, equation (3.2) is satisfied for all k and the average is conserved. Moreover, we can consider a vector $\mathbf{u}_k \in \mathbb{R}^m$ as the control variable that represents the weights on the undirected links (each link $s \sim (ij)$ is given a control $u_s^{(k)}$). At stage k, the network designer will select a control $\mathbf{u}_k$. In particular, due to equation (3.4) we can write the weight matrix as a function of the control vector as follows:

$$W(k) = I_n - Q\text{diag}(\mathbf{u}_k)Q^T. \tag{3.5}$$

For any iteration k, the square error L_k metric measures the distance of the system from the average. Since the goal is to reach faster the consensus fast, cost is assigned only to the last stage. The optimal control problem of this chapter is then given as follows:

$$\begin{aligned} &\underset{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}}{\operatorname{argmin}} \quad L_N \\ &\text{subject to} \\ &\qquad \mathbf{y}_{k+1} = \mathbf{y}_k - Q\operatorname{diag}(\mathbf{u}_k)Q^T\mathbf{y}_k, \text{ for } k = 0,\ldots,N-1, \end{aligned} \tag{3.6}$$

where N is the number of stages in this optimization. Let us first show that an optimal control $(\mathbf{u}_k^*,\ k = 0,\ldots,N-1)$ that solves the optimization problem (3.6) exists.

3.2.1 Existence of a Solution

The cost function of the optimization problem is given by

$$\begin{aligned} L_N &= \mathbf{y}_N^T\mathbf{y}_N \\ &= \mathbf{x}(N)^T\mathbf{x}(N) - 2\bar{\mathbf{x}}^T\mathbf{x}(N) + \bar{\mathbf{x}}^T\bar{\mathbf{x}} \\ &= \mathbf{x}(N)^T\mathbf{x}(N) - 2x_{ave}\mathbf{1}^T\mathbf{x}(N) + nx_{ave}^2 \\ &= J_N - nx_{ave}^2, \end{aligned}$$

where $J_N = \mathbf{x}(N)^T\mathbf{x}(N)$. Then minimizing $L_N = \mathbf{y}_N^T\mathbf{y}_N$ is equivalent to minimizing the function $J_N = \mathbf{x}(N)^T\mathbf{x}(N)$ because the term nx_{ave}^2 depends only on the initial values. Let us define the product matrix $U_{(k_1:k_2)}$ as follows:

$$U_{(k_1:k_2)} = \begin{cases} W(k_1)W(k_1+1)\ldots W(k_2) & \text{if } k_1 < k2 \\ W(k_1)W(k_1-1)\ldots W(k_2) & \text{if } k_1 > k2 \\ W(k_1) & \text{if } k_1 = k2. \end{cases}$$

To show that an optimal control $(\mathbf{u}_k^*,\ k = 0,\ldots,N-1)$ exists, we write the optimization as an unconstrained one:

$$\underset{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}}{\operatorname{argmin}} f(\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}) \tag{3.7}$$

where

$$\begin{aligned} f(\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}) = J_N &= \mathbf{x}(N)^T\mathbf{x}(N) \\ &= \mathbf{x}(0)^T U_{(N-1,0)}^T U_{(N-1,0)}\mathbf{x}(0). \end{aligned} \tag{3.8}$$

Since the elements of the matrix $U_{(N-1,0)}$ are linear in the control variables, and $U_{(N-1,0)}^T U_{(N-1,0)}$ is a positive semi-definite matrix, $f(.)$ is a quadratic function and bounded from below, and hence there exists at least one control $(\mathbf{u}_k^*,\ k = 0,\ldots,N-1)$ that globally minimizes f.

3.2.2 Necessary Conditions

To find necessary conditions for the optimal control, we apply the maximum principle [LVS12, p. 24] to problem (3.6). For $k = 0, \ldots, N-1$, the system equation, performance index, and Hamiltonian are given as:

- System equation:
$$\mathbf{y}_{k+1} = \mathbf{y}_k - Q\text{diag}(\mathbf{u}_k)Q^T\mathbf{y}_k, \tag{3.9}$$
- Performance index:
$$L_N = \mathbf{y}_N^T\mathbf{y}_N,$$
- Hamiltonian:
$$H^k = \lambda_{k+1}^T \left(\mathbf{y}_k - Q\text{diag}(\mathbf{u}_k)Q^T\mathbf{y}_k\right), \tag{3.10}$$
where λ_{k+1} is the costate variable corresponding to iteration k.

Then, the costate equation and the associated boundary condition are:

- Costate equation:
$$\lambda_k = \frac{\partial H^k}{\partial \mathbf{y}_k} = \left(I_n - Q\text{diag}(\mathbf{u}_k)Q^T\right)\lambda_{k+1}, \tag{3.11}$$
- Boundary condition:
$$\lambda_N = \mathbf{y}_N. \tag{3.12}$$

Any optimal control should minimize the Hamiltonian [LVS12]. Since the Hamiltonian is linear in the *unconstrained* control variables, if any coefficient of a control variable in (3.10) is nonzero, the optimal control would be unbounded. But an optimal control exists as we have already shown, so all the coefficients of the control variables in (3.10) are necessarily equal to zero, i.e.,

$$\frac{\partial H^k}{\partial \mathbf{u}_k} = \left(Q^T\mathbf{y}_k\right) \odot \left(Q^T\lambda_{k+1}\right) = \mathbf{0}, \quad \text{for } k = 0, \ldots, N-1, \tag{3.13}$$

where $\odot$ is the element-wise product of the vectors and $\mathbf{0}$ is the vector of all zeros. Equation (3.13) provides necessary conditions for a controller to minimize (3.8).

These necessary conditions can be further simplified giving a simple network interpretation if we consider one stage ($N = 1$). In fact, using the boundary condition (3.12), the necessary conditions in equation (3.13) for $N = 1$ reduce to

$$\left(Q^T\mathbf{y}_0\right) \odot \left(Q^T\mathbf{y}_1\right) = \mathbf{0},$$

i.e., for any link $(ij) \in E$ we have

$$(x_i(0) - x_j(0))(x_i(1) - x_j(1)) = 0. \tag{3.14}$$

Let $H = (V, E')$ be a sub-graph of G defined on the same set of vertices, V, and with links $E' \subseteq E$ such that $(ij) \in E'$ if $(ij) \in E$ and $x_i(0) - x_j(0) \neq 0$. Then we have the following:

Proposition 6. *If $H = (V, E')$ is connected, then any optimal control $\mathbf{u}^*$ drives the system to consensus in one iteration, i.e.,*

$$\bar{\mathbf{x}} = \left(I_n - Q\,diag(\mathbf{u}^*)Q^T\right)\mathbf{x}(0).$$

Proof. Due to Eq. (3.14), we have $x_i(1) = x_j(1)\ \ \forall (ij) \in E'$. If H is connected, then there is a path in E' between any two vertices, and thus $x_i(1) = x_j(1)\ \forall i, j \in V$. Using also the fact that the average is conserved (due to equation (3.2)), we get $x_i(1) = x_{ave}\ \ \forall i \in V$. □

3.2.3 Locally Optimal Solution

In the general case, the optimization problem (3.7) is computationally hard because the function $f(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1})$ is not convex (it is convex in the variables of each stage, $\mathbf{u}_k$, but not jointly convex). We therefore turn our attention to locally optimal solutions, and to obtain such a solution we apply the gradient method to (3.8).

Proposition 7. *Let $f(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1})$ be given by (3.8). Then, for $k = 0, \ldots, N-1$, the gradient $g_l^{(k)}$ of the function $f(.)$ with respect to its variables $u_l^{(k)}$ where $u_l^{(k)}$ is the l-th element of the vector $\mathbf{u}_k$ corresponding to link (ij) ($l \sim (ij)$) at stage k, is given as follows:*

$$\begin{aligned} g_l^{(k)} &= \frac{\partial f}{\partial u_l^{(k)}} \\ &= 2[(A_k W(k) B_k)_{ij} + (A_k W(k) B_k)_{ji} \\ &\qquad - (A_k W(k) B_k)_{ii} - (A_k W(k) B_k)_{jj}], \end{aligned} \tag{3.15}$$

where A_k and B_k are as follows:

$$\begin{aligned} A_k &= \begin{cases} U_{(N-1:k+1)}^T U_{(N-1:k+1)} & \text{if } N-1 \geq k+1, \\ I_n & \text{if } N-1 < k+1, \end{cases} \\ B_k &= \begin{cases} \left(U_{(k-1:0)}\mathbf{x}(0)\right)\left(U_{(k-1:0)}\mathbf{x}(0)\right)^T & \text{if } k-1 \geq 0, \\ \mathbf{x}(0)\mathbf{x}(0)^T & \text{if } k-1 < 0. \end{cases} \end{aligned} \tag{3.16}$$

Proof. By using the commutative property of the trace operator (i.e., $Tr(XY) = Tr(YX)$ for any conformable matrices X and Y), $f(.)$ can be written for any $k = 0, \ldots, N-1$ as follows:

$$\begin{aligned} f(\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}) &= \mathbf{x}(0)^T U_{(N-1,0)}^T U_{(N-1,0)} \mathbf{x}(0) \\ &= Tr\left(W(k)^T A_k W(k) B_k\right), \end{aligned} \tag{3.17}$$

where A_k and B_k are given by (3.16) and are independent of the variables of stage k (i.e., $\frac{\partial (A_k)_{st}}{\partial u_l^{(k)}} = \frac{\partial (B_k)_{st}}{\partial u_l^{(k)}} = 0\ \ \forall s, t \in V,$ and $k = 0, \ldots, N-1$). From matrix calculus,

if $h(W) = Tr(W^T AWB)$, then $\frac{\partial h}{\partial w_{ij}} = 2\,(AWB)_{ij}$, and since $W = I_n - Q\text{diag}(\mathbf{u})Q^T$, for any u_l such that $l \sim (ij)$ we have

$$\frac{\partial w_{st}}{\partial u_l} = \begin{cases} +1 & \text{if } s = i \text{ and } t = j \\ +1 & \text{if } s = j \text{ and } t = i \\ -1 & \text{if } s = i \text{ and } t = i \\ -1 & \text{if } s = j \text{ and } t = j \\ 0 & \text{else.} \end{cases} \tag{3.18}$$

Thus,

$$\begin{aligned} \frac{\partial h}{\partial u_l} &= \sum_{s,t} \left(\frac{\partial h}{\partial w_{st}} \right) \frac{\partial w_{st}}{\partial u_l} = 2 \sum_{s,t} (AWB)_{st} \frac{\partial w_{st}}{\partial u_l} \\ &= 2\left[(AWB)_{ij} + (AWB)_{ji} - (AWB)_{ii} - (AWB)_{jj} \right]. \end{aligned} \tag{3.19}$$

We can apply equation (3.19) to every stage separately and this ends the proof. □

Let us stack up all the elements $u_l^{(k)}$ in one vector $\mathbf{w}$, and also stack up all the elements $g_l^{(k)}$ in one vector $\mathbf{g}$.

Proposition 8. *Consider the following gradient iterative procedure*

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \gamma_t \mathbf{g}^{(t)},$$

where $\gamma_t = \frac{1}{(1+t)||\mathbf{g}||}$ *is the stepsize and* $\mathbf{w}^{(0)} = \mathbf{0}$. *Then the elements* $u_l^{(k)}$ *of the vector* $\mathbf{w}$ *converge to a locally minimizing solution of the optimization problem* (3.6).

Proof. The given procedure is a standard (sub-)gradient method for optimization and the convergence has been widely studied under the diminishing step-size rule: $\lim_{t\to\infty} \gamma_t = 0$ and $\sum_{t=1}^{\infty} \gamma_t = \infty$ (see [Sho85]). □

Remark: The function $f(.)$ can have multiple local minima, and the gradient method converges to one of them. But simulations show that in some situations any local minimum is in fact a global one. This is the case when we start with initial values where only one node i has a nonzero estimate $x_i(0) = 1$, and all other nodes have an initial value 0. If d is the largest distance (in terms of the number of hops) from node i to a node j having $x_j(0) = 0$, then we know that any optimal control needs at least d stages to drive the system to consensus because node j needs at least d iterations to change its value 0. By simulations, the gradient given in Proposition 8 for $N = d$ stages yields weights that lead to consensus (as we will see later in Section 3.4) and hence the solution turns out to be optimal (global minimum).

3.2.4 Closed-Form Solution for the One-Stage Problem

Consider now the case $N = 1$, that is with only one stage. Then the control would be a single vector $\mathbf{u}$ where each component is the weight for the corresponding edge. The optimization problem in this case is convex:

$$\mathbf{u}_S = \underset{\mathbf{u}}{\operatorname{argmin}} \quad f(\mathbf{u}), \tag{3.20}$$

where $\mathbf{u}_S$ is the solution set (possibly an infinite set) and

$$\begin{aligned} f(\mathbf{u}) &= \mathbf{x}(0)^T(I_n - Q\text{diag}(\mathbf{u})Q^T)(I_n - Q\text{diag}(\mathbf{u})Q^T)\mathbf{x}(0) \\ &= ||\mathbf{x}(0) - Q\text{diag}(\mathbf{u})Q^T\mathbf{x}(0)||^2 \\ &= ||\mathbf{x}(0) - Q\text{diag}(Q^T\mathbf{x}(0))\mathbf{u}||^2 \\ &= ||D\mathbf{u} - \mathbf{b}||^2, \end{aligned}$$

where

$$D = Q\text{diag}(Q^T\mathbf{x}(0)), \text{ and } \mathbf{b} = \mathbf{x}(0). \tag{3.21}$$

The problem is then reduced to a least squares approximation problem, where any element in the solution set $\mathbf{u}_S$ satisfies what is known as the normal equations:

$$D^TD\mathbf{u} = D^T\mathbf{b}, \quad \forall \mathbf{u} \in \mathbf{u}_S.$$

Moreover, $\mathbf{u}_S$ is not empty, with at least one solution $\hat{\mathbf{u}}$,

$$\hat{\mathbf{u}} = D^+\mathbf{b},$$

where D^+ is the pseudo inverse of D that can be obtained using the singular value decomposition of D. If D^TD is a positive definite matrix, then $D^+ = (D^TD)^{-1}D^T$ and $\hat{\mathbf{u}}$ is the unique solution to the least squares problem. We denote by S the minimum value of the function $f(\mathbf{u})$:

$$\begin{aligned} S &= f(\hat{\mathbf{u}}) \\ &= ||(DD^+ - I)\mathbf{b}||^2. \end{aligned} \tag{3.22}$$

3.3 Network with Adversary in Discrete Time

Suppose that there is an adversary that can add noise onto the weights of the links. The adversary's objective is to drive the system away from consensus. Considering only one stage optimization ($N = 1$), the state equation would become

$$\begin{aligned} \mathbf{x}(1) &= W(\mathbf{u}, \mathbf{v})\mathbf{x}(0) \\ &= (I_n - Q\text{diag}(\mathbf{u} + \mathbf{v})Q^T)\mathbf{x}(0), \end{aligned} \tag{3.23}$$

where $W(\mathbf{u}, \mathbf{v})$ is the weight matrix that depends on the control $\mathbf{u} \in U_1 = \mathbb{R}^m$ and the noise of the adversary $\mathbf{v} \in U_2 = \{y; y \in \mathbb{R}^m, ||y|| \leq C\}$, where C is a given positive constant and can be seen as the power constraint of the adversary (the larger C the more powerful is the adversary). The cost function is

$$\begin{aligned} J(\mathbf{u}, \mathbf{v}) &= \mathbf{x}(1)^T\mathbf{x}(1) \\ &= ||(I_n - Q\text{diag}(\mathbf{u} + \mathbf{v})Q^T)\mathbf{x}(0)||^2 \\ &= ||D(\mathbf{u} + \mathbf{v}) - \mathbf{b}||^2, \end{aligned} \tag{3.24}$$

where D and $\mathbf{b}$ are given by (3.21). The adversary ($\mathbf{v}$) is the maximizer of $J(\mathbf{u}, \mathbf{v})$ while the network designer ($\mathbf{u}$) is the minimizer in this zero-sum two-person game.

Definition 1. *A pair* $(\mathbf{u}^* \in U_1, \mathbf{v}^* \in U_2)$ *is a saddle-point in pure strategies of* $J(\mathbf{u}, \mathbf{v})$ *if the following holds:*

$$J(\mathbf{u}^*, \mathbf{v}) \leq J(\mathbf{u}^*, \mathbf{v}^*) \leq J(\mathbf{u}, \mathbf{v}^*), \text{ for all } (\mathbf{u} \in U_1, \mathbf{v} \in U_2).$$

The *lower value* $\underline{V}$ and the *upper value* $\overline{V}$ of the game are defined by

$$\underline{V} = \sup_{\mathbf{v} \in U_2} \inf_{\mathbf{u} \in U_1} J(\mathbf{u}, \mathbf{v}) \text{ , and } \overline{V} = \inf_{\mathbf{u} \in U_1} \sup_{\mathbf{v} \in U_2} J(\mathbf{u}, \mathbf{v}).$$

Since the strategy spaces are decoupled, $\underline{V} \leq \overline{V}$. If furthermore $\underline{V} = \overline{V}$, then the common value is called the *value* of the game. Existence of a saddle-point guarantees existence of the value [BO99]. As J is a quadratic function of $\mathbf{u}$, and $J(\mathbf{u}, \mathbf{v}) \geq 0$ for all $(\mathbf{u} \in U_1, \mathbf{v} \in U_2)$, then for any given $\mathbf{v} \in U_2$, J attains a minimum on U_1 [Hil08]. Moreover, since U_2 is compact, and J is a continuous function on its domain of definition, for any given $\mathbf{u} \in U_1$, J attains a maximum on U_2 by the Weierstrass Theorem. Therefore, we can replace $\inf_{\mathbf{u} \in U_1}$ by $\min_{\mathbf{u} \in U_1}$ and $\sup_{\mathbf{v} \in U_2}$ by $\max_{\mathbf{v} \in U_2}$ in the definitions of the upper and lower values. In the sequel, we will show that actually the game does not have a value, and hence does not have a saddle-point (in pure strategies). It however has a saddle-point in mixed strategies (shortly to be defined).

3.3.1 The max-min Solution

In the max-min solution, the network designer has access to the strategy played by the adversary.

$$\begin{aligned} \operatorname*{argmin}_{\mathbf{u}} J(\mathbf{u}, \mathbf{v}) &= \operatorname*{argmin}_{\mathbf{u}} ||D(\mathbf{u} + \mathbf{v}) - \mathbf{b}||^2 \\ &= D^+\mathbf{b} - \mathbf{v}. \end{aligned}$$

Then we have,

$$\begin{aligned} \max_{\mathbf{v}} \min_{\mathbf{u}} J(\mathbf{u}, \mathbf{v}) &= \max_{\mathbf{v}} J(D^+\mathbf{b} - \mathbf{v}, \mathbf{v}) \\ &= \max_{\mathbf{v}} S \\ &= S, \end{aligned} \tag{3.25}$$

where S is the value of the one player optimization problem, given by (3.22) and is independent of $\mathbf{v}$.

3.3.2 The min-max Solution

In the min-max solution, the adversary has access to the strategy of the controller. The cost function J can be written as:

$$\begin{aligned} J(\mathbf{u}, \mathbf{v}) &= ||D(\mathbf{u} + \mathbf{v}) - \mathbf{b}||^2 \\ &= \mathbf{b}^T\mathbf{b} + \mathbf{u}^T D^T D\mathbf{u} - 2\mathbf{b}^T D\mathbf{u} + \mathbf{v}^T D^T D\mathbf{v} + 2\mathbf{v}^T \left(D^T D\mathbf{u} - D^T\mathbf{b}\right). \end{aligned}$$

Consider the following strategy $\mathbf{v}_1$ by the adversary:

$$\begin{cases} \mathbf{v}_1 \in \mathcal{R}(D^TD) \cap U_2 & \text{if } D^TD\mathbf{u} - D^T\mathbf{b} = \mathbf{0} \\ \mathbf{v}_1 = C\frac{(D^TD\mathbf{u} - D^T\mathbf{b})}{||D^TD\mathbf{u} - D^T\mathbf{b}||} & \text{otherwise,} \end{cases} \tag{3.26}$$

where $\mathcal{R}(D^TD)$ is the range of the matrix D^TD. Therefore we have,

$$\begin{aligned} \min_{\mathbf{u}} \max_{\mathbf{v}} J(\mathbf{u}, \mathbf{v}) &\geq \min_{\mathbf{u}} J(\mathbf{u}, \mathbf{v}_1) \\ &= \min_{\mathbf{u}} \Bigg(\underbrace{\mathbf{v}_1^T D^T D \mathbf{v}_1 + 2\mathbf{v}_1^T \left(D^TD\mathbf{u} - D^T\mathbf{b}\right)}_{>0} \\ &\qquad + \mathbf{b}^T\mathbf{b} + \mathbf{u}^T D^T D\mathbf{u} - 2\mathbf{b}^T D\mathbf{u} \Bigg) \\ &> \min_{\mathbf{u}} \left(\mathbf{b}^T\mathbf{b} + \mathbf{u}^T D^T D\mathbf{u} - 2\mathbf{b}^T D\mathbf{u}\right) \\ &= S. \end{aligned}$$

Hence,

$$\max_{\mathbf{v}} \min_{\mathbf{u}} J(\mathbf{u}, \mathbf{v}) < \min_{\mathbf{u}} \max_{\mathbf{v}} J(\mathbf{u}, \mathbf{v}), \tag{3.27}$$

which means that there is no saddle-point in pure strategies.

3.3.3 A Saddle-Point Equilibrium (SPE) in Mixed Strategies

Since an SPE does not exist in pure strategies, we allow players to randomize their actions through mixed strategies. A mixed strategy for the network designer is a probability distribution μ on U_1, and we denote the space of all such probability distributions by M_1. Similarly, a mixed strategy for the adversary is a probability distribution ν on U_2, and the space of all such probability distributions is denoted by M_2. The average cost corresponding to a pair $(\mu \in M_1, \nu \in M_2)$ is given by

$$\bar{J}(\mu, \nu) = \int_{U_1 \times U_2} J(\mathbf{u}, \mathbf{v}) d\mu(\mathbf{u}) d\nu(\mathbf{v}).$$

Definition 2. *A pair $(\mu^* \in M_1, \nu^* \in M_2)$ is a saddle-point equilibrium in mixed strategies if the following holds:*

$$\bar{J}(\mu^*, \nu) \leq \bar{J}(\mu^*, \nu^*) \leq \bar{J}(\mu, \nu^*), \text{ for all } (\mu \in M_1, \nu \in M_2).$$

Proposition 9. *Consider the following strategies:*

$$\mu^*(\mathbf{u}) : \mathbf{u} = D^+\mathbf{b} \text{ with probability 1,} \tag{3.28}$$

and

$$\nu^*(\mathbf{v}) : \begin{cases} \mathbf{v} = C\mathbf{p} & \text{with probability 1/2} \\ \mathbf{v} = -C\mathbf{p} & \text{with probability 1/2,} \end{cases} \tag{3.29}$$

where $\mathbf{p}$ is any unit eigenvector of the matrix D^TD corresponding to the largest eigenvalue $\lambda_{\max}(D^TD)$. Then the pair (μ^*, ν^*) is an SPE in mixed strategies.

Proof. Let us recall the cost function:

$$\begin{aligned} J(\mathbf{u},\mathbf{v}) &= \mathbf{b}^T\mathbf{b} + \mathbf{u}^T D^T D\mathbf{u} - 2\mathbf{b}^T D\mathbf{u} \\ &\quad + \mathbf{v}^T D^T D\mathbf{v} + 2\mathbf{v}^T \left(D^T D\mathbf{u} - D^T\mathbf{b}\right) \\ &= ||D\mathbf{u} - \mathbf{b}||^2 + \mathbf{v}^T D^T D\mathbf{v} + 2\mathbf{v}^T \left(D^T D\mathbf{u} - D^T\mathbf{b}\right). \end{aligned}$$

Then the average cost under the given pair of strategies is,

$$\begin{aligned} \bar{J}(\mu^*,\nu^*) &= ||DD^+\mathbf{b} - \mathbf{b}||^2 + (C\mathbf{p})^T D^T D(C\mathbf{p}) \times (1/2) \\ &\quad + (-C\mathbf{p})^T D^T D(-C\mathbf{p}) \times (1/2) \\ &= S + C^2\lambda_{\max}. \end{aligned} \tag{3.30}$$

But we have,

$$\begin{aligned} \bar{J}(\mu^*,\nu) &= ||DD^+\mathbf{b} - \mathbf{b}||^2 + \int_{U_2} \nu^T D^T D\nu \; \mathrm{d}\nu(\mathbf{v}) \\ &\leq S + \max_{\mathbf{v},||\mathbf{v}||\leq C} \mathbf{v}^T D^T D\mathbf{v} \\ &= S + C^2\lambda_{\max} \\ &= \bar{J}(\mu^*,\nu^*), \end{aligned} \tag{3.31}$$

and

$$\begin{aligned} \bar{J}(\mu,\nu^*) &= C^2\lambda_{\max} + \int_{U_1} ||D\mu - \mathbf{b}||^2 \; \mathrm{d}\mu(\mathbf{u}) \\ &\geq C^2\lambda_{\max} + \min_{\mathbf{u}} ||D\mathbf{u} - \mathbf{b}||^2 \\ &= S + C^2\lambda_{\max} \\ &= \bar{J}(\mu^*,\nu^*). \end{aligned} \tag{3.32}$$

Since we have for any pair $(\mu \in M_1, \nu \in M_2)$,

$$\bar{J}(\mu^*,\nu) \leq \bar{J}(\mu^*,\nu^*) \leq \bar{J}(\mu,\nu^*),$$

then (μ^*,ν^*) is a saddle-point equilibrium. □

Remark: The saddle-point is not unique, as any (μ,ν) where μ is a point distribution in the set $\mathbf{u}_S$ of (3.20) (or any distribution on this set due to the interchangeability property of saddle-points [BO99]), and ν as in (3.29) where $\mathbf{p}$ is any eigenvector corresponding to $\lambda_{\max}(D^T D)$ (or any distribution on these vectors) is also a saddle-point. However, if D is full column rank, and $\lambda_{\max}$ has geometric multiplicity of 1, then the saddle-point is unique.

3.4 Simulations

3.4.1 Optimal Control

We illustrate the results obtained on a numerical example. Given the sample network of Fig. 3.1 and the initial values, we are interested in selecting the controls on links,

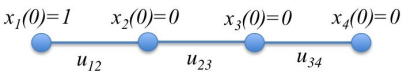

Figure 3.1: Network with 4 communicating nodes. $x_i(0)$ is the initial value of node i, and u_{ij} is the control value (or weight) of link (ij).

$k=0$		$k=1$	
$\mathbf{x}(0)$	$\mathbf{u}_0^*$	$\mathbf{x}(1)$	$\mathbf{u}_1^*$
$\begin{pmatrix}1\\0\\0\\0\end{pmatrix}$	$\begin{pmatrix}0.8665\\0\\0\end{pmatrix}$	$\begin{pmatrix}0.1335\\0.8665\\0\\0\end{pmatrix}$	$\begin{pmatrix}0.2201\\0.6051\\0\end{pmatrix}$
$J_0 = \mathbf{x}(0)^T\mathbf{x}(0) = 1$		$J_1 = 0.7686$	
$k=2$		$k=3$	
$\mathbf{x}(2)$	$\mathbf{u}_2^*$	$\mathbf{x}(3)$	
$\begin{pmatrix}0.2949\\0.1808\\0.5243\\0\end{pmatrix}$	$\begin{pmatrix}0.3934\\0.0708\\0.4768\end{pmatrix}$	$\begin{pmatrix}0.25\\0.25\\0.25\\0.25\end{pmatrix}$	
$J_2 = 0.3945$		$J_3 = 0.25$	

Table 3.1: Optimal control results for the network in Fig. 3.1.

$\mathbf{u}_k = (u_{12}^{(k)}, u_{23}^{(k)}, u_{34}^{(k)})^T$, so that the system reaches consensus. We limit the number of stages to $N = 3$ because in that case the diameter is equal to three and an optimal control that drives the system to consensus exists. The optimization problem (3.6) reduces to:

$$\begin{aligned}&\underset{\mathbf{u}_0,\mathbf{u}_1,\mathbf{u}_2}{\text{argmin}} \quad J_3\\ &\text{subject to}\\ &\qquad \mathbf{x}(k+1) = (I_4 - Q\text{diag}(\mathbf{u}_k)Q^T)\mathbf{x}(k), \text{ for } k = 0,1,2,\end{aligned}$$

where I_4 is the 4×4 identity matrix, and the incidence matrix Q is given by:

$$Q = \begin{pmatrix}1 & 0 & 0\\ -1 & 1 & 0\\ 0 & -1 & 1\\ 0 & 0 & -1\end{pmatrix}.$$

Table 3.1 shows the optimal control $(\mathbf{u}_k^*, k = 0,1,2)$ for the given network. The control is obtained by the gradient descent iterative procedure of Proposition 8 where the initial starting point of the gradient was selected 0 on all links of the three stages. The results indicate that with only three iterations, the system reaches consensus. To compare with other weight selection algorithms, we apply the algorithm given in

[XB04] obtained for a related semi-definite program (SDP). That algorithm finds a fixed set of weights for all iterations that guarantee fastest convergence independent of initial values (worst-case analysis). For the network example in Fig. 3.1, the SDP assigns a value 0.5 to all weights for all iterations, and the resulting state vector after three iterations is $\mathbf{x}_{SDP}(3) = (0.375, 0.375, 0.125, 0.125)^T$, which has a cost of $J_3 = 0.3125$ (thus higher cost than our time-varying weights) and needs an infinite number of iterations to converge. It is worth mentioning that the SDP weights are designed for worst-case node initial values, and thus have the advantage that they guarantee convergence starting from any initial values. However, the optimal control in this chapter is designed for a given starting value, and thus if the initial node values change, the control values must be readjusted.

3.4.2 Adversarial Intervention

In this subsection, we study the effect of an adversary disrupting the communication on networks with connected random geometric graphs (RGGs) topology where n nodes are thrown uniformly at random on a unit square, and any two nodes within a connectivity radius r are connected by a link (the simulations are done with $r = \sqrt{0.6 \times \frac{\log(n)}{n}}$ given that the graph is connected). RGGs are generally used as models for wireless sensor networks, and disruption of communication can be accomplished by insertion of high intensity signals on communication links. The additive white noise can also be considered as an adversarial input in our settings. We compare the results on different RGGs with different sizes (number of nodes n) for $n \in \{20, 40, 60, 80, 100\}$. Fig. 3.2 depicts the different costs on the resulting network with and without the presence of the adversary, averaged over 150 independent runs. We consider only one-stage games where the initial cost function is given by $J_0 = \mathbf{x}(0)^T\mathbf{x}(0)$. For any node i, the initial node value $x_i(0)$ is selected at random uniformly within the interval $[0, 1]$. We assume that the adversary power constraint is $||\mathbf{v}|| \leq 1$ (i.e., $C = 1$). We see from Fig. 3.2 that the network without an adversary achieves the least cost J_1. An adversary selecting uniformly random strategy from the n-dimensional unit sphere does not substantially affect the cost; however, an adversary with the same power constraint playing the strategy of the saddle-point equilibrium (equation (3.29)) achieves significantly higher cost than the uniform random adversary (even larger cost than J_0 for graphs of $n = 20$ and $n = 40$ nodes).

3.5 Conclusion

In this chapter, we have studied a finite-horizon discrete-time optimal control problem for a network designer to achieve faster consensus given the network structure and the initial node values. The optimal control is obtained using gradient methods. We have also provided sufficient conditions for reaching consensus in one stage. Moreover, we have studied the saddle-point equilibrium (SPE) of the consensus problem in the presence of an adversary, and found that an SPE does not exist in

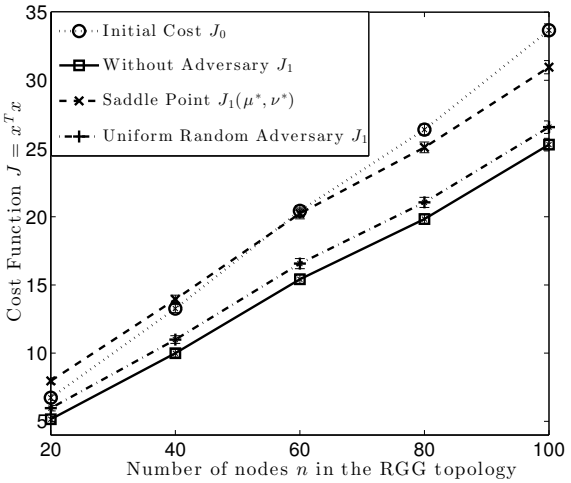

Figure 3.2: The cost function due to different adversary settings: absence of adversary, uniform random adversary that adds a random noise to the control values, and saddle-point adversary that randomizes its strategy in accordance with the saddle-point equilibrium.

pure strategies. Nevertheless, an SPE exists in mixed strategies, where the adversary selects the noise using a randomized strategy, whereas the network designer's strategy is still pure.

Chapter 4

Quantized Communication in Consensus Protocols

Contents

Most existing algorithms (as well as the ones we've seen so far) for precise distributed averaging require that agents are able to send and receive real values with infinite precision. However, a realistic network can only allow messages with limited length to be transmitted between agents due to constraints on the capacity of communication links. With such a constraint, when a real value is sent from an agent to its neighbors, this value will be truncated and only a quantized version will be received by the neighbors. With such quantization, the precise average cannot be achieved (except in particular cases), but some value close to it can be achieved, called quantized consensus. A number of papers have studied this quantized consensus problem and various *probabilistic* quantization strategies have been proposed to cause all the agents in a network to reach a quantized consensus with probability one (or at least with high probability)

[Sch64, AB10, ACR07, BTV09, BTV11, LM12, KM10, KBS07, EB13]. Notwithstanding this, the problem of how to design and analyze *deterministic* quantization effects remains open [FCFZ09, CYRC13].

In this chapter, we thoroughly analyze the performance of distributed averaging algorithms where the information exchange between neighboring agents is subject to a deterministic uniform quantization. We show that in finite time, the algorithm will either cause all agents to reach a quantized consensus where the consensus value is the largest integer not greater than the average of their initial values, or will lead all agents' variables to cycle in a small neighborhood around the average, depending on initial conditions. In the latter case, we give tight error bounds for the size of the neighborhood and it is further shown that the error can be made arbitrarily small by adjusting the algorithm's parameters in a distributed manner, at a cost of slower convergence.

4.1 Literature Review

Most of the related works for distributed averaging with quantized communication use either a deterministic algorithm (as our approach in this chapter) or a probabilistic one.

There are only a few publications which study deterministic algorithms for quantized consensus. In [LFXZ11] the distributed averaging problem with quantized communication is formulated as a feedback control design problem for coding/decoding schemes; the paper characterizes the amount of information needed to be sent for the agents to reach a consensus and shows that with an appropriate scaling function and some carefully chosen control gain, the proposed protocol can solve the distributed averaging problem, but some spectral properties of the Laplacian matrix of the underlying fixed undirected graph have to be known in advance. More sophisticated coding/decoding schemes were proposed in [LX11] for time-varying undirected graphs and in [ZZ13] for time-varying directed graphs, all requiring carefully chosen parameters. Recently a novel dynamic quantizer has been proposed in [TKPF13] based on dynamic quantization intervals for coding of the exchanged messages in wireless sensor networks leading to asymptotic convergence to consensus. In [CM09] a biologically inspired algorithm was proposed which makes all agents reach some consensus with arbitrary precision, but at the cost of not preserving the desired average. Control performance of logarithmic quantizers was studied in [CFSZ08] and quantization effects were considered in [NOOT09]. A deterministic algorithm of the same form as in this chapter has been only partially analyzed in [FCFZ09] where the authors have approximated the system by a probabilistic model and left the design of the weights as an open problem.

Over the past decade quite a few probabilistic quantized consensus algorithms have been proposed. The probabilistic quantizer in [ACR07] ensures almost sure consensus at a common but random quantization level for fixed (strongly connected) directed graphs; although the expectation of the consensus value equals the desired

average, the deviation of the consensus value from the desired average is not tightly bounded. An alternative algorithm which gets around this limitation was proposed in [KM10]; the algorithm adds dither to the agents' variables before quantization and the mean square error can be made arbitrarily small by tuning the parameters. The probabilistic algorithm in [BTV09, BTV11], called "interval consensus gossip", causes all agents to reach a consensus in finite time almost surely on the interval in which the average lies, for time-varying (jointly connected) undirected graphs. Stochastic quantized gossip algorithms were introduced in [LM12, ZM11] and shown to work properly. The effects of quantized communication on the standard randomized gossip algorithm [BGPS06] were analyzed in [CFFZ10]. An alternative approach to analyze the quantization effect was introduced in [Sch64, AB10] which model the effect as noise following certain probability.

Another thread of research has studied quantized consensus with the additional constraint that the value at each node is an integer. The probabilistic algorithm in [KBS07] causes all agents to reach quantized consensus almost surely for a fixed (connected) undirected graph; convergence time of the algorithm was studied in [EB13], with strong bounds on its expected value. In [CI11] a probabilistic algorithm was proposed to solve the quantized consensus problem for fixed (strongly connected) directed graphs using the idea of "surplus".

We should note that, in addition, our work in this chapter is also related to the literature on the problem of load balancing [AAMR93, SS94, GM96].

4.2 System Equation

In this chapter, we will refer to the nodes running the distributed averaging as agents. As assumed so far, the graph G is connected and does not change over time. Initially each agent i has a real number $x_i(0)$. Let

$$x_{ave}(k) = \frac{1}{n}\sum_{i\in V} x_i(k),$$

be the average of values of all agreement variables in the network, x_{ave} is then simply $x_{ave}(0)$. The approach studied so far in this thesis to the problem is for each agent to use a linear iterative update rule of the form

$$x_i(k+1) = w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k), \quad \forall i \in V. \tag{4.1}$$

4.3 Quantized Communication

In a network where links have constraints on the capacity and have limited bandwidth (e.g., digital communication networks), messages cannot have infinite length. However, the distributed averaging algorithm requires sending real (infinite precision) values through these communication links. Therefore, with digital transmission, the messages transmitted between neighboring agents will have to be truncated. If the communication bandwidth was limited, the more the truncation of

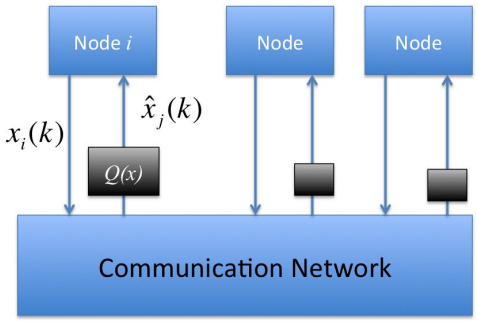

Figure 4.1: The network model for the quantized system.

agents' values, the higher would be the deviation of agent's value from the desired average consensus x_{ave}.

To model the effect of quantized communication, we assume that the links perform a quantization effect on the values transmitted between agents. The network model is given by Fig. 4.1. As we can see from the model, each agent i can have infinite bandwidth to store its latest value $x_i(k)$ and perform computations. However, when agent i sends its value at time k through the communication network, its neighbors will receive a value $\hat{x}_i(k)$ which is the quantized value of $x_i(k)$. A *quantizer* is a function $\mathcal{Q} : \mathbb{R} \to \mathbb{Z}$ that maps a real value to an integer. Quantizers can be of different forms. We present here some widely used quantizers in the literature [NFZE07, CFFZ10, NOOT09]:

1. Truncation quantizer $\mathcal{Q}_t$ which truncates the decimal part of a real number and keeps the integer part:

$$\mathcal{Q}_t(x) = \lfloor x \rfloor. \tag{4.2}$$

2. Ceiling quantizer $\mathcal{Q}_c$ which rounds the value to the nearest upper integer:

$$\mathcal{Q}_c(x) = \lceil x \rceil. \tag{4.3}$$

3. Rounding quantizer $\mathcal{Q}_r$ which rounds a real number to its nearest integer:

$$\mathcal{Q}_r(x) = \begin{cases} \lfloor x \rfloor & \text{if } x - \lfloor x \rfloor < 1/2 \\ \lceil x \rceil & \text{if } x - \lfloor x \rfloor \geq 1/2. \end{cases} \tag{4.4}$$

4. Probabilistic quantizer $\mathcal{Q}_p$ defined as follows:

$$\mathcal{Q}_p(x) = \begin{cases} \lfloor x \rfloor & \text{with probability } \lceil x \rceil - x \\ \lceil x \rceil & \text{with probability } x - \lfloor x \rfloor. \end{cases} \tag{4.5}$$

In this chapter we study the effect of the deterministic quantizers ($\mathcal{Q}_t(x)$, $\mathcal{Q}_c(x)$, and $\mathcal{Q}_r(x)$) on the performance of the distributed averaging algorithms by showing the distance that the agents' stored values can deviate from the initial average x_{ave}. The quantizers listed before map $\mathbb{R}$ into $\mathbb{Z}$ and have quantization jumps of size 1. Quantizers having a generic real positive quantization step ϵ can be simply recovered by a suitable scaling: $\mathcal{Q}^{(\epsilon)}(x) = \epsilon\mathcal{Q}(x/\epsilon)$ [CFFZ10]. Thus the results in this chapter cover these generic quantizers as well.

4.4 Problem Formulation

Suppose that all n agents adhere to the same update rule of Eq. (4.1). Then with a quantizer $\mathcal{Q}(x)$, the network equation would be

$$x_i(k+1) = w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}\mathcal{Q}(x_j(k)), \quad \forall i \in V. \tag{4.6}$$

Simple examples show that this algorithm can cause the system to shift away from the initial average x_{ave}.

Since agents know exactly the effect of the quantizer, for the agents not to lose any information caused by quantization, at each iteration k each agent i can send out the quantized value $\mathcal{Q}(x_i(k))$ (instead of sending $x_i(k)$) and store in a *local* scalar $c_i(k)$ the difference between the real value $x_i(k)$ and its quantized version, i.e.,

$$c_i(k) = x_i(k) - \mathcal{Q}(x_i(k)).$$

Then, the next iteration update of agent i can be modified to be

$$x_i(k+1) = w_{ii}\mathcal{Q}(x_i(k)) + \sum_{j\in N_i} w_{ij}\mathcal{Q}(x_j(k)) + c_i(k), \quad \forall i \in V. \tag{4.7}$$

A major difference between this equation and (4.6) is that here no information is lost; i.e., the total average is being conserved in the network, as we will show shortly after. The state equation of the system becomes,

$$\mathbf{x}(k+1) = W\mathcal{Q}\left(\mathbf{x}(k)\right) + \mathbf{x}(k) - \mathcal{Q}\left(\mathbf{x}(k)\right), \tag{4.8}$$

where, with a little abuse of notation, $\mathcal{Q}\left(\mathbf{x}\right) = \left(\mathcal{Q}(x_1), \mathcal{Q}(x_2), \ldots, \mathcal{Q}(x_n)\right)^T$ is the vector quantization operation. For any W where each column sums to 1 ($\mathbf{1}^T W = \mathbf{1}^T$ where $\mathbf{1}$ is the vector of all ones), the total sum of all n agreement variables does not change over time if agents followed the protocol of Eq. (4.8):

$$\begin{aligned}
\mathbf{1}^T\mathbf{x}(k+1) &= \mathbf{1}^T(W\mathcal{Q}\left(\mathbf{x}(k)\right) + \mathbf{1}^T\mathbf{x}(k) - \mathbf{1}^T\lfloor\mathbf{x}(k)\rfloor \\
&= \mathbf{1}^T\mathcal{Q}\left(\mathbf{x}(k)\right) + \mathbf{1}^T\mathbf{x}(k) - \mathbf{1}^T\mathcal{Q}\left(\mathbf{x}(k)\right) \\
&= \mathbf{1}^T\mathbf{x}(k) \\
&= \mathbf{1}^T\mathbf{x}(0) \\
&= nx_{ave},
\end{aligned} \tag{4.9}$$

Thus the average is also conserved ($x_{ave}(k) = x_{ave}, \quad \forall k$). Equation (4.8) would be our model of distributed averaging with deterministic quantized communication where the quantizer can take the form of the truncation $\mathcal{Q}_t$, the ceiling $\mathcal{Q}_c$, or the rounding one $\mathcal{Q}_r$. It is worth noting that the three quantizers can be related by the following equations:

$$\mathcal{Q}_r(x) = \mathcal{Q}_t(x + 1/2), \tag{4.10}$$

$$\mathcal{Q}_c(x) = -\mathcal{Q}_t(-x). \tag{4.11}$$

Given a model with the ceiling quantizer $\mathcal{Q}_c$ in (4.8), by taking $\mathbf{y}(k) = -\mathbf{x}(k)$, the system evolves as:

$$\begin{aligned} \mathbf{y}(k+1) &= \mathbf{y}(k) + W\mathcal{Q}_t(\mathbf{y}(k)) - \mathcal{Q}_t(\mathbf{y}(k)) \\ \mathbf{y}(0) &= -\mathbf{x}(0). \end{aligned}$$

Therefore, by analyzing the above system which has a truncation quantizer $\mathcal{Q}_t$, we can deduce the performance of $\mathbf{x}(k)$ that satisfies equation (4.8) with a ceiling quantizer $\mathcal{Q}_c$ because they are related by a simple equation ($\mathbf{y}(k) = -\mathbf{x}(k)$).

Similarly, given a model with the rounding quantizer $\mathcal{Q}_r$ in (4.8), by taking $\mathbf{y}(k) = \mathbf{x}(k) + \frac{1}{2}\mathbf{1}$, the system evolves as:

$$\begin{aligned} \mathbf{y}(k+1) &= \mathbf{y}(k) + W\mathcal{Q}_t(\mathbf{y}(k)) - \mathcal{Q}_t(\mathbf{y}(k)) \\ \mathbf{y}(0) &= \mathbf{x}(0) + \frac{1}{2}\mathbf{1}. \end{aligned}$$

Therefore, by analyzing the above system which has a truncation quantizer $\mathcal{Q}_t$, we can deduce the performance of $\mathbf{x}(k)$ that satisfies equation (4.8) with a rounding quantizer $\mathcal{Q}_r$ because they are related by a simple translation equation ($\mathbf{y}(k) = \mathbf{x}(k) + \frac{1}{2}\mathbf{1}$). Therefore the effects of all these three quantizers are essentially the same.

With this nontrivial observation in mind, we focus on the analysis of the truncation quantizer only in the rest of this chapter. The results can then be easily extended to the case of the other two quantizers.

In the sequel we will fully characterize the behavior of system (4.8) and its convergence properties. But first, we have the following definition:

Definition 3. *A network of n agents reaches quantized consensus if there is an iteration k_0 such that*

$$\mathcal{Q}(x_i(k)) = \mathcal{Q}(x_j(k)), \quad \forall i, j \in V, \ \forall k \geq k_0.$$

4.5 Design and Analysis of the System

In this section, we carry out the analysis of the proposed quantized system equation. By considering the truncation quantizer $\mathcal{Q}_t$ in (4.8), the system equation becomes:

$$\mathbf{x}(k+1) = W\lfloor \mathbf{x}(k) \rfloor + \mathbf{x}(k) - \lfloor \mathbf{x}(k) \rfloor. \tag{4.12}$$

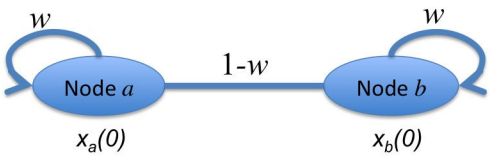

Figure 4.2: Network of two nodes where quantized communication does not converge.

This can be written in a distributed way for every $i \in V$ as follows:

$$x_i(k+1) = x_i(k) + \sum_{j \in N_i} w_{ji} \left(\lfloor x_j(k) \rfloor - \lfloor x_i(k) \rfloor \right), \tag{4.13}$$

$$= x_i(k) + \sum_{j \in N_i} w_{ji} L_{ji}(k), \tag{4.14}$$

where

$$L_{ji}(k) \triangleq \lfloor x_j(k) \rfloor - \lfloor x_i(k) \rfloor = -L_{ij}(k).$$

The non-linearity of the system due to quantization complicates the analysis, and traditional stability analysis of linear systems (such as ergodicity, products of stochastic matrices, etc.) cannot be applied here as the system might not even converge. As demonstrated in the following subsection.

4.5.1 Cyclic Example

The purpose of the following example is to show that for a "bad" weight matrix design, the quantized system can cycle very far from the average. Consider the two-nodes example of Fig. 4.2, suppose that $x_a(0) = \xi$, $x_b(0) = K + \xi$ where $K \in \mathbb{N}$ and $\xi \in (0, 1)$. With these initial values, $\lfloor x_a(0) \rfloor = 0$, $\lfloor x_b(0) \rfloor = K$, and $x_{ave} = \frac{K}{2} + \xi$. The weight matrix for this two-nodes system is assumed to be a doubly stochastic matrix and is given as follows:

$$W = \begin{pmatrix} w & 1-w \\ 1-w & w \end{pmatrix},$$

where $w \in (0, 1)$. With this weight matrix, (4.9) is satisfied and the average is conserved. In [FCFZ09], the authors defined the following metric to measure the performance of the system:

$$d_\infty(W, \mathbf{x}(0)) = \limsup_{k \to \infty} \frac{1}{\sqrt{n}} ||\Delta(k)||, \tag{4.15}$$

where $\Delta(k)$ is a vector having the elements $\Delta_i(k) = x_i(k) - x_{ave}$. So the worst cycle (according to this metric), given a doubly stochastic weight matrix, would happen if the nodes toggled their values with every iteration. Let us derive conditions on

W for which this could happen. With the quantization, the corresponding system equations are as follows:

$$x_a(k+1) = x_a(k) + (1-w) \times (\lfloor x_b(k) \rfloor - \lfloor x_a(k) \rfloor) \tag{4.16}$$
$$x_b(k+1) = x_b(k) + (1-w) \times (\lfloor x_a(k) \rfloor - \lfloor x_b(k) \rfloor)\,. \tag{4.17}$$

From the given initial conditions, after one iteration the updated values are $x_a(1) = \xi + (1-w)K$ and $x_b(1) = K + \xi - (1-w)K$. Therefore, the quantized value of the nodes' variables will toggle between 0 and K if $x_a(1) \in [K, K+1)$ and $x_b(1) \in [0,1)$. By substituting the values of $x_a(1)$ and $x_b(1)$ we get the following necessary conditions for such a cycle,

$$\begin{cases} wK > \max\{-\xi, \xi - 1\} \\ wK < \min\{\xi, 1-\xi\}. \end{cases} \tag{4.18}$$

The first condition is always satisfied because $wK > 0$. Then, a bad design of W is to have $w < \frac{1}{K} \times \min\{\xi, 1-\xi\}$ because in this case the nodes can cycle[1] with

$$x_a(k) = \begin{cases} \xi & \text{if } k \text{ is even} \\ K + \xi - wK & \text{if } k \text{ is odd} \end{cases} \quad \text{and} \quad x_b(k) = \begin{cases} K + \xi & \text{if } k \text{ is even} \\ wK + \xi & \text{if } k \text{ is odd.} \end{cases} \tag{4.19}$$

Thus $\Delta_a(k) = \Delta_b(k) = K/2$ if k is even, and so $d_\infty(W, \mathbf{x}(0)) = K/2$. The above two-node network result can be extended to regular bipartite graphs where the first set of nodes takes the value $x_a(0)$ and the other set takes the value $x_b(0)$ and all self-weights are equal to w.[2] This would also lead to the following inequality on $d_\infty(W, \mathbf{x}(0))$ with the given initial conditions and weight matrix:

$$d_\infty(W, \mathbf{x}(0)) \geq K/2.$$

This shows that a bad design of W on general graphs can make the cycle arbitrarily large.

4.5.2 Weight Assumption

The system behavior depends of course on the design of the weight matrix. In distributed averaging, it is important to consider weights that can be chosen locally, avoid *bad* design, and guarantee desired convergence properties. We impose the following assumption on W which can be satisfied in a distributed manner.

Assumption 1. *The weight matrix in our design has the following properties:*

[1] In case initial values were not known, since $\min\{\xi, 1-\xi\} \leq 1/2$, then, a bad design of W is to have $w < \frac{1}{2K}$ because in this case there might be some initial values that cause large cycles.

[2] In case of hypercube graphs, [FCFZ09] shows that if the weights in the network have a constant value $1/(d+1)$ where $d = \log n$ is the degree of a node in the hypercube graph, then an upper bound on $d_\infty(W) = \sup_{\mathbf{x}(0)} d_\infty(W, \mathbf{x}(0))$ is the following $d_\infty(W) \leq \frac{\log n}{2}$. Since a hypercube is a regular bipartite graph, then using our results leads to the following lower bound, $d_\infty(W) \geq \frac{\log n}{4}$ (by taking $\xi = 0.5$ and $K = (\log n)/2$ to satisfy (4.18)).

- *W is a symmetric doubly stochastic matrix:*

$$w_{ij} = w_{ji} \geq 0 \quad \forall i, j \in V$$
$$\sum_i w_{ij} = \sum_j w_{ij} = 1,$$

- *Dominant diagonal entries of W: $w_{ii} > 1/2$ for all $i \in V$,*
- *Network communication constraint: if $(i, j) \notin E$, then $w_{ij} = 0$,*
- *For any link $(i, j) \in E$ we have $w_{ij} \in \mathbb{Q}^+$, where $\mathbb{Q}^+$ is the set of rational numbers in the interval $(0, 1)$.*

These are also sufficient conditions for the linear system (4.1) to converge. The restriction of the weights to the class of rational numbers is just because of a technical reason to prove convergence results.

We now state the main result of this chapter which will be proved in the following subsections.

Main Convergence Result 1. *Consider the quantized system (4.12). Suppose that Assumption 1 holds. Then for any initial value $\mathbf{x}(0)$, there is a finite time iteration where either*

1. *the system reaches quantized consensus, or*
2. *the nodes' values cycle in a small neighborhood around the average, where the neighborhood can be made arbitrarily small by a decentralized design of the weights (having trade-off with the speed of convergence).*

To highlight the importance of these results, notice that the Main Convergence Result 1 implies there is an iteration k_0 such that $x_i(k) - x_j(k) < 1$ for all $i, j \in V$ for $k \geq k_0$. This gives a constant upper bound on the metric $d_\infty(W) = \sup_{\mathbf{x}(0)} d_\infty(W, \mathbf{x}(0))$ independent of initial values, i.e., due to Assumption 1, $d_\infty(W) \leq 0.5$ on any general graph and for any initial conditions.

4.5.3 Cyclic States

We study in this subsection the convergence properties of the system equation (4.12) under Assumption 1. Let us first show that due to quantized communication, the states of the agents lie in a discrete set. Since $w_{ij} \in \mathbb{Q}^+$ for any link (i, j), we can write

$$w_{ij} = \frac{a_{ij}}{b_{ij}},$$

where a_{ij} and b_{ij} are co-prime positive integers. Suppose that B_i is the Least Common Multiple (LCM) of the integers $\{b_{ij}; (i, j) \in E, j \in N_i\}$. Let $c_i(k) =$

$x_i(k) - \lfloor x_i(k) \rfloor$; then we have $c_i(k) \in [0, 1)$. Let us see how $c_i(k)$ evolves:

$$\begin{aligned} c_i(k) &= x_i(k) - \lfloor x_i(k) \rfloor \\ &= x_i(k-1) + \sum_{j \in N_i} w_{ij} \times (\lfloor x_j(k-1) \rfloor - \lfloor x_i(k-1) \rfloor) \\ &\quad - \lfloor x_i(k) \rfloor \\ &= \lfloor x_i(k-1) \rfloor + c_i(k-1) \\ &\quad + \sum_{j \in N_i} \frac{a_{ij}}{b_{ij}} \times (\lfloor x_j(k-1) \rfloor - \lfloor x_i(k-1) \rfloor) - \lfloor x_i(k) \rfloor \\ &= c_i(k-1) + \frac{Z(k)}{B_i}, \end{aligned} \tag{4.20}$$

where $Z(k) \in \mathbb{Z}$ is an integer. Then with a simple recursion, we can see that for any iteration k we have:

$$c_i(k) = c_i(0) + \frac{\tilde{Z}(k)}{B_i}, \tag{4.21}$$

where $\tilde{Z}(k) \in \mathbb{Z}$. Since $c_i(k) \in [0, 1)$, this equation shows that the states of the nodes are quantized, and the decimal part can have maximum B_i quantization levels.

We now give the following definition,

Definition 4. *The quantized system (4.12) is cyclic if there exists a positive integer P and a finite time k_0 such that*

$$\mathbf{x}(k+P) = \mathbf{x}(k) \quad \forall k \geq k_0,$$

where P is the cycle period.

Proposition 10. *Suppose Assumption 1 holds. Then, the quantized system (4.12), starting from any initial value $\mathbf{x}(0)$, is cyclic.*

Proof. Let $m(k)$ and $M(k)$ be defined as follows:

$$m(k) \triangleq \min_{i \in V} \lfloor x_i(k) \rfloor, \quad M(k) \triangleq \max_{i \in V} \lfloor x_i(k) \rfloor. \tag{4.22}$$

Notice that for any k, we have

$$\begin{aligned} x_i(k+1) &= x_i(k) + \sum_{j \in N_i} w_{ji} L_{ji} \\ &\leq c_i(k) + \lfloor x_i(k) \rfloor + \left(\sum_{j \in N_i} w_{ji} \right) (M(k) - \lfloor x_i(k) \rfloor) \\ &\leq c_i(k) + M(k), \end{aligned}$$

from which it follows that $\lfloor x_i(k+1) \rfloor \leq M(k)$, and hence $M(k+1) \leq M(k)$. By a simple recursion we can see that the maximum cannot increase, $M(k) \leq M(0)$. Similarly, we have $m(k) \geq m(0)$.

As a result, $\lfloor x_i(k) \rfloor \in \{m(0), m(0)+1, \ldots, M(0)-1, M(0)\}$ is a finite set. Moreover, from equation (4.21), $c_i(k)$ belongs to a finite set that can have at most B_i elements. Since $x_i(k) = \lfloor x_i(k) \rfloor + c_i(k)$, and each of the elements in the sum belongs to a finite set, $x_i(k)$ belongs to a finite set as well (of maximum cardinality $B_i(M(0) - m(0) + 1)$). But from equation (4.12), we have $\mathbf{x}(k+1) = f(\mathbf{x}(k))$ where the function $f(.)$ is a deterministic function of the input state at iteration k, so the system is a deterministic finite state automata. Since the system is deterministic, it would enter a cycle if the same state is reached at two different iterations. The total number of states is upper bounded by $D = (B(M(0) - m(0) + 1))^n$ where $B = \max_i B_i$, and the system enters a cycle in finite time $T \leq D$ because if $T > D$, then at least one state is repeated. □

4.5.4 Lyapunov Stability

In this subsection, we will study the stability of the above system using a Lyapunov function. Assumption 1 and Eq. (4.21) imply that there exists a fixed[3] strictly positive constant $\gamma > 0$ such that for any i and any iteration k the following hold:

$$\text{If } c_i(k) > \left(\sum_{j \in N_i} w_{ij} \right), \text{ then } \quad c_i(k) - \sum_{j \in N_i} w_{ij} \geq 2\gamma, \tag{4.23}$$

$$\text{If } \bar{c}_i(k) > \left(\sum_{j \in N_i} w_{ij} \right), \text{ then } \quad \bar{c}_i(k) - \sum_{j \in N_i} w_{ij} \geq 2\gamma, \tag{4.24}$$

$$\bar{c}_i(k) \geq 2\gamma, \tag{4.25}$$

$$\frac{1}{2} - \sum_{j \in N_i} w_{ij} \geq 2\gamma, \tag{4.26}$$

where $\bar{c}_i(k) = 1 - c_i(k)$. Let $\gamma_{\max}$ be the maximum γ that satisfies equations (4.23)-(4.26). The results thereafter hold for any $\gamma \in (0, \gamma_{\max}]$.

Remark: *Equations (4.23)-(4.25) do not hold for the simple linear model of (4.1). For example, consider a linear model that does not reach consensus in finite time, and suppose that $x_{ave} \in \mathbb{Z}$. Then, since $\lim_{k\to\infty} x_i(k) = x_{ave}$, we have that $c_i(k)$ can be as close to 1 as desired, and hence we cannot bound $\bar{c}_i(k)$ by a fixed positive value.*

In fact, equations (4.23)-(4.25) show the discrete nature of the quantized system where $c_i(k)$ can only take finite possible values. We will use these equations to define a closed interval (set) $I = [a, b]$ having the property that if $x_i(k) \in I$, then $x_i(k)$ is an interior point in this interval with a distance at least γ far from its boundaries. Having a fixed distance γ from the boundaries will play an important role in the stability analysis in what follows because it shows that if a node's variable got out of the interval, it must pass at least a distance γ, i.e., suppose that $x_i(k) \in I$

[3] By 'fixed' we mean that the value is independent of time and it only depends on initial values and the network structure.

but $x_i(k+1) \notin I$, then $d(x_i(k+1), I) \leq |x_i(k+1) - x_i(k)| - \gamma$ where $d(x, I) = \min_{y \in I} |x - y|$ is the distance of the node's variable x from the interval I.

Let $m(k)$ and $M(k)$ be defined as in (4.22). Let us define the following set:

$$S_k = \{\mathbf{y} \in \mathbb{R}^n, |y_i - m(k) - 1| \leq \alpha_i \text{ for all } i\}, \tag{4.27}$$

where $\alpha_i = 1 - w_{ii} + \gamma$. Note that

$$\begin{aligned} \alpha_i &= 1 - w_{ii} + \gamma \\ &= \sum_{j \in N_i} w_{ij} + \gamma \\ &\leq \frac{1}{2} - \gamma, \end{aligned}$$

where the last inequality is due to Eq. (4.26), and thus $\alpha_i \in (0, 1/2)$. The set S_k depends on the iteration k because the value m does. Since according to the system (4.12), $m(k)$ cannot decrease and $M(k)$ cannot increase as indicated earlier, then S_k can only belong to one of the $M(0) - m(0)$ possible compact sets at each iteration k. Furthermore, if S_k changes to a different compact set due to an increase in m, it cannot go back to the old one as m cannot decrease. Additionally, if $\mathbf{x}(k) \in S_k$, then it is an interior point of the set S_k and not on the boundary because suppose $|x_i(k) - m(k) - 1| = \alpha_i$, then either $c_i(k) = \alpha_i = \sum_{j \in N_i} w_{ij} + \gamma$ which contradicts (4.23) or $\bar{c}_i(k) = \alpha_i = \sum_{j \in N_i} w_{ij} + \gamma$ which contradicts (4.24).

Let us define the following candidate Lyapunov function:

$$\begin{aligned} V(k) &= d(\mathbf{x}(k), S_k) \\ &= \min_{\mathbf{y} \in S_k} ||\mathbf{y} - \mathbf{x}(k)||_1 \\ &= \min_{\mathbf{y} \in S_k} \sum_{i \in V} |y_i - x_i(k)| \end{aligned} \tag{4.28}$$

By minimizing along each component of $\mathbf{y}$ independently, we get

$$V(k) = \sum_i \max\{|x_i(k) - m(k) - 1| - \alpha_i, 0\}.$$

Let us determine the change in the proposed candidate Lyapunov function. In order to understand the evolution of $\nabla V_k = V(k+1) - V(k)$, we group the nodes depending on their values at iteration k into 6 sets, $X_1(k)$, $X_2(k)$, $X_3(k)$, $X_4(k)$, $X_5(k)$, and $X_6(k)$ (see Fig. 4.3):

- Node $i \in X_1(k)$ if $\quad m(k) \leq x_i(k) < m(k) + 1 - \alpha_i$,
- Node $i \in X_2(k)$ if $\quad m(k) + 1 - \alpha_i \leq x_i(k) < m(k) + 1$,
- Node $i \in X_3(k)$ if $\quad m(k) + 1 \leq x_i(k) \leq m(k) + 1 + \alpha_i$,
- Node $i \in X_4(k)$ if $\quad m(k) + 1 + \alpha_i < x_i(k) < m(k) + 2$,

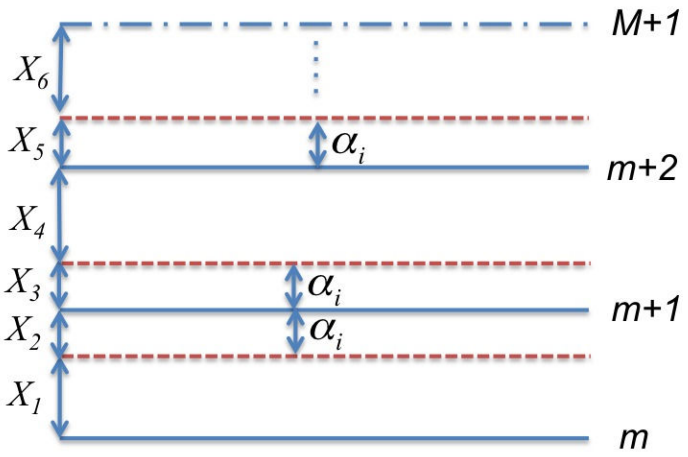

Figure 4.3: Dividing the nodes into sets according to their local values.

- Node $i \in X_5(k)$ if $\quad m(k) + 2 \leq x_i(k) < m(k) + 2 + \alpha_i$,
- Node $i \in X_6(k)$ if $\quad m(k) + 2 + \alpha_i \leq x_i(k)$.

For simplicity we will drop the index k in the notation of the sets and $m(k)$ when there is no confusion. To have better insights about these sets, we note that if X_6 becomes empty at a given iteration, then the set remains empty, i.e.,

Lemma 5. *If $X_6(k_0) = \phi$, then $X_6(k) = \phi$ for all $k \geq k_0$.*

Proof. If a node $i \notin X_6(k)$, then $\lfloor x_i(k) \rfloor \in \{m(k), m(k)+1, m(k)+2\}$. So for any node i,

$$\begin{aligned} x_i(k+1) &= x_i(k) + \sum_{j \in N_i} w_{ij} L_{ji} \\ &< m(k) + 2 + \alpha_i \end{aligned}$$

where the last equality is due to three possibilities,

- if $\lfloor x_i(k) \rfloor = m(k)+2$, then $L_{ji} \leq 0$ for every $j \in N_i$, and $x_i(k) < m(k)+2+\alpha_i$ since $i \in X_5$ in this case;
- if $\lfloor x_i(k) \rfloor = m(k)+1$, then $\sum_{j \in N_i} w_{ij} L_{ji} \leq \sum_{j \in N_i \cap X_5} w_{ij} \leq \alpha_i$, and $x_i(k) < m(k)+2$ in this case;
- if $\lfloor x_i(k) \rfloor = m(k)$, then $\sum_{j \in N_i} w_{ij} L_{ji} \leq \sum_{j \in N_i} w_{ij} \times 2 \leq 2\alpha_i$, and $x_i(k) < m(k)+1$ in this case.

Therefore, since $m(k) \leq m(k+1)$, then $x_i(k+1) < m(k+1)+2+\alpha_i$ and $i \notin X_6(k+1)$ from the definition of the sets and this ends the proof. □

Note that by a similar reasoning as in Lemma 5, if $\{X_5, X_6\}$ got empty, then it remains empty during all further iterations, and if $\{X_4, X_5, X_6\}$ got empty it remains empty too.

With every iteration, nodes can change their sets. Note that any node can jump in one iteration to a higher set, but the other way around is not always possible. For example, a node at iteration k in X_1 can jump at iteration $k+1$ to X_6, but no node outside X_1 can get back to it (if the minimum $m(k)$ is not increased) as we will show next.

Lemma 6. *If $m(k+1) = m(k)$ and $i \notin X_1(k)$, then $i \notin X_1(k+1)$.*

Proof. Let us define L_i^k be the level of node i at iteration k, i.e., $L_i^k = \lfloor x_i(k) \rfloor - m(k)$. Then,

$$\begin{aligned} x_i(k+1) &= x_i(k) + \sum_{j \in N_i} w_{ji} L_{ji} \\ &\geq c_i(k) + \lfloor x_i(k) \rfloor + (\sum_{j \in N_i} w_{ji})(m(k) - \lfloor x_i(k) \rfloor) \\ &= c_i(k) + L_i^k + m(k) + (\sum_{j \in N_i} w_{ji})(-L_i^k) \\ &= m(k) + c_i(k) + w_{ii} L_i^k \\ &\geq m(k) + 1 - \alpha_i \\ &\geq m(k+1) + 1 - \alpha_i, \end{aligned}$$

and $i \notin X_1(k+1)$. The inequality before the last one is due to two possibilities,

- if $i \in X_2(k)$ then $L_i^k = 0$, and $m(k) + c_i(k) = x_i(k) \geq m(k) + 1 - \alpha_i$,
- otherwise $L_i^k \geq 1$, so $m(k) + c_i(k) + w_{ii} L_i^k \geq m(k) + w_{ii} \geq m(k) + 1 - \alpha_i$.

□

Therefore, due to Lemma 6 the increase $V(k)$ is due to nodes changing to a higher set. However, any node changing its set to a higher one, should have neighbors in the higher sets that cause $V(k)$ to decrease by at least the same amount. To make this a formal argument we give the following lemma:

Lemma 7. *Consider the quantized system (4.12). Suppose that Assumption 1 holds. If $m(k+1) = m(k)$, we have*

$$\nabla V_k \leq 0.$$

Proof. We define $\nabla_i V_k$ as follows:

$$\begin{aligned} \nabla_i V_k \triangleq \max\{|x_i(k+1) - m - 1| - \alpha_i, 0\} \\ - \max\{|x_i(k) - m - 1| - \alpha_i, 0\}, \end{aligned} \tag{4.29}$$

from which it is evident that $\nabla V_k = \sum_{i \in V} \nabla_i V_k$. Since only nodes moving from a set X_s to a higher set X_t where $t \geq \max\{s, 4\}$ can increase $V(k)$ (we will use the expression $X_s \rightarrow X_t$ to denote the transition of a node that belongs to the set X_s at iteration k to the set X_t at iteration $k+1$), then we can enumerate all the possible transitions of nodes that can cause $V(k)$ to increase:

1. $X_1(k) \rightarrow X_t(k+1)$, $t \geq 4$,

$$
\begin{aligned}
\nabla_i V_k &= \max\{|x_i(k+1) - m - 1| - \alpha_i, 0\} - \max\{|x_i(k) - m - 1| - \alpha_i, 0\} \\
&= (x_i(k+1) - m - 1 - \alpha_i) - (1 + m - x_i(k) - \alpha_i) \\
&= x_i(k) + \sum_{j \in N_i} w_{ij} \left(\lfloor x_j(k) \rfloor - \lfloor x_i(k) \rfloor \right) - m - 1 - m - 1 + x_i(k) \\
&= \sum_{j \in N_i} w_{ij} L_{ji} - 2(m + 1 - x_i(k)) \\
&= \sum_{j \in N_i} w_{ij} L_{ji} - 2\bar{c}_i(k) \\
&= \sum_{j \in N_i} w_{ij} L_{ji} - 2(\alpha_i(k) - \alpha_i(k) + \bar{c}_i(k)) \\
&= \Big(\sum_{j \in N_i \cap \{X_3, X_4\}} w_{ij} \Big) + \Big(\sum_{j \in N_i \cap X_5} w_{ij} \times 2 \Big) + \Big(\sum_{j \in N_i \cap X_6} w_{ij} L_{ji} \Big) \\
&\quad - 2\Big(\sum_{j \in N_i} w_{ij} + \gamma + (\bar{c}_i(k) - \alpha_i) \Big) \\
&\leq \underbrace{\Big(\sum_{j \in N_i \cap X_6} w_{ij} L_{ji} \Big)}_{\geq 0} - 4\gamma.
\end{aligned} \tag{4.30}
$$

2. $X_2(k) \rightarrow X_t(k+1)$, $t \geq 4$, and the change in the Lyapunov function due to

these nodes is as follows:

$$\begin{aligned}
\nabla_i V_k &= \max\{|x_i(k+1) - m - 1| - \alpha_i, 0\} \\
&\quad - \max\{|x_i(k) - m - 1| - \alpha_i, 0\} \\
&= (x_i(k+1) - m - 1 - \alpha_i) - 0 \\
&= x_i(k) + \sum_{j\in N_i} w_{ij}L_{ji} - m - 1 - \alpha_i \\
&= \sum_{j\in N_i} w_{ij}L_{ji} - \alpha_i - \bar{c}_i(k) \\
&= (\sum_{j\in N_i\cap\{X_3,X_4\}} w_{ij}) + (\sum_{j\in N_i\cap X_5} w_{ij}\times 2) \\
&\quad + (\sum_{j\in N_i\cap X_6} w_{ij}L_{ji}) - \sum_{j\in N_i} w_{ij} - \gamma - \bar{c}_i(k) \\
&\le \underbrace{(\sum_{j\in N_i\cap X_5} w_{ij})}_{\ge 0} + \underbrace{(\sum_{j\in N_i\cap X_6} w_{ij}L_{ji})}_{\ge 0} - 2\gamma.
\end{aligned} \tag{4.31}$$

3. $X_3(k) \to X_t(k+1)$, $t \ge 4$, then

$$\begin{aligned}
\nabla_i V_k &= x_i(k) + \sum_{j\in N_i} w_{ij}L_{ji} - m - 1 - \alpha_i \\
&= \sum_{j\in N_i} w_{ij}L_{ji} - (\alpha_i - c_i(k)) \\
&= (\sum_{j\in N_i\cap\{X_1,X_2\}} w_{ij}\times(-1)) + (\sum_{j\in N_i\cap X_5} w_{ij}) \\
&\quad + (\sum_{j\in N_i\cap X_6} w_{ij}L_{ji}) - (\alpha_i - c_i(k)) \\
&\le \underbrace{(\sum_{j\in N_i\cap X_5} w_{ij})}_{\ge 0} + \underbrace{(\sum_{j\in N_i\cap X_6} w_{ij}L_{ji})}_{\ge 0} - \gamma.
\end{aligned} \tag{4.32}$$

4. $X_4(k) \to X_t(k+1)$, $t \ge 4$, then

$$\begin{aligned}
\nabla_i V_k &= \sum_{j\in N_i} w_{ij}L_{ji} \\
&\le \underbrace{\left(\sum_{j\in N_i\cap X_5} w_{ij}\right)}_{\ge 0} + \underbrace{\left(\sum_{j\in N_i\cap X_6} w_{ij}L_{ji}\right)}_{\ge 0}.
\end{aligned}$$

5. $X_5(k) \to X_t(k+1)$, $t \geq 5$, then

$$\begin{aligned}\nabla_i V_k &= \sum_{j \in N_i} w_{ij} L_{ji} \\ &= \underbrace{\left(\sum_{j \in N_i \cap X_6} w_{ij} L_{ji} \right)}_{\geq 0} + \underbrace{\left(\sum_{j \in N_i, j \notin X_6} w_{ij} L_{ji} \right)}_{\leq 0}.\end{aligned}$$

6. $X_6(k) \to X_6(k+1)$, then

$$\begin{aligned}\nabla_i V_k &= \sum_{j \in N_i} w_{ij} L_{ji} \\ &= \underbrace{\left(\sum_{j \in N_i \cap \bar{X}_6^i} w_{ij} L_{ji} \right)}_{\geq 0} + \underbrace{\left(\sum_{j \in N_i, j \notin \bar{X}_6^i} w_{ij} L_{ji} \right)}_{\leq 0}.\end{aligned}$$

where the set $\bar{X}_6^i$ is the set of nodes such that $j \in \bar{X}_6^i$ if $x_j(k) \geq x_i(k)$.

Notice that the positive component in ∇V_k because of a node s belonging to one of the presented 6 possibilities is only due to a neighbor p in $\{X_5(k), X_6(k)\}$ such that $x_p(k) \geq x_s(k)$. Then p can belong to two possible sets: X_5 or X_6.

Suppose first that $p \in X_6(k)$, let A be the increase in $\nabla_s V_k$, then this increase is as follows:

$$A = w_{ps} L_{ps} > 0,$$

but this increase is decreased again in $\nabla_p V_k$ since a node in $X_6(k)$ cannot drop below $X_4(k+1)$, we can write:

$$\begin{aligned}\nabla_p V_k &= \max\{|x_p(k+1) - m - 1| - \alpha_p, 0\} \\ &\quad - \max\{|x_p(k) - m - 1| - \alpha_p, 0\} \\ &= (x_p(k+1) - m - 1 - \alpha_p) - (x_p(k) - 1 - m - \alpha_p) \\ &= x_p(k) + \sum_{j \in N_p} w_{jp} L_{jp} - x_p(k) \\ &= \underbrace{w_{sp} L_{sp}}_{-A} + \sum_{j \in N_p - \{s\}} w_{jp} L_{jp}.\end{aligned}$$

Taking the other case, suppose now $p \in X_5$, let B be the increase in $\nabla_s V_k$ of a node s due to its neighbor $p \in X_5$:

$$B = w_{sp} > 0,$$

then this increase is decreased again in $\nabla_p V_k$, but we should consider two cases:

- p: $X_5 \to X_m$, $m \geq 4$, then

$$\nabla_p V_k = \underbrace{w_{ps} L_{sp}}_{\leq -B} + \sum_{j \in N_p - \{s\}} w_{jp} L_{jp}, \tag{4.33}$$

- p: $X_5 \to X_3$, then

$$\begin{aligned} \nabla_p V_k &\leq -1/2 \\ &\leq -\sum_{j \in N_p} w_{pj} \\ &= \underbrace{-w_{ps}}_{-B} - \sum_{j \in N_p - \{s\}} w_{jp}, \end{aligned}$$

and p decreases in the same amount that its neighbor s increased.

Remark: *For every positive value that increases $V(k)$, there is a* unique *corresponding negative value that compensates this increase by decreasing $V(k)$. This is because for any link $l \sim (i,j) \in E$, the increase in $\nabla_i V_k$ due to l forces a decrease in $\nabla_j V_k$ due to the same link, and so there is one to one mapping between the increased values and the decreased ones.*

As a result of the discussion we can have the total ∇V_k cannot increase, namely

$$\nabla V_k = \sum_i \nabla_i V_k \leq 0.$$

□

Lemma 7 implies that $V(k)$ is non-increasing with time. We identify some situations under which $V(k)$ strictly decreases (assuming of course $m(k+1) = m(k)$). Given for example a node $i \in X_1(k)$ that is connected to a node $j \in \{X_3(k), X_4(k), X_5(k), X_6(k)\}$, if i jumped to $X_t(k+1), t \geq 4$, then the term -4γ from equation (4.30) causes strict decrease in $V(k)$, i.e., $\nabla V_k \leq -4\gamma$. If $i \in X_t(k+1), t < 4$, then $x_i(k+1) = x_i(k) + w_{ij}(\lfloor x_s(k) \rfloor - m) + \sum_{s \in N_i - j} w_{is}(\lfloor x_s(k) \rfloor - m) \geq x_i(k) + w_{ij}$ and thus

$$\nabla_i V_k \leq -\min\{w_{ij}, d(x_i(k), [m+1-\alpha_i, m+1+\alpha_i])\} \leq -\min\{\delta, \gamma\},$$

where $d(x_i(k), [m+1-\alpha_i, m+1+\alpha_i])$ is the distance of $x_i(k)$ from the set $[m+1-\alpha_i, m+1+\alpha_i]$ and $\delta = \min_{(i,j) \in E} w_{ij} > 0$. This decrease in $\nabla_i V_k$ causes ∇V_k to decrease by the same quantity. Another situation can arise if, for example, a node $i \in X_2(k)$ is connected to a node $j \in \{X_4(k), X_5(k), X_6(k)\}$. If i jumped to $X_t(k+1), t \geq 4$, then the term -2γ from equation (4.31) causes a strict decrease in $V(k)$, i.e., $\nabla V_k \leq -2\gamma$. If $i \in X_t(k+1), t < 4$ (and so is any neighbor in $\{X_2(k), X_3(k)\}$ of j), then $x_j(k+1) = x_j(k) + w_{ij}(\lfloor x_i(k) \rfloor - \lfloor x_j(k) \rfloor) + \sum_{s \in N_j - i} w_{sj}(\lfloor x_s(k) \rfloor - \lfloor x_j(k) \rfloor) \leq x_j(k) - w_{ij} + \sum_{s \in N_j - i} w_{sj}(\lfloor x_s(k) \rfloor - \lfloor x_j(k) \rfloor)$ and thus a term $-\min\{w_{ij}, d(x_j(k), [m+1-\alpha_j, m+1+\alpha_j])\}$ appears in $\nabla_j V_k$ which causes $\nabla V_k \leq -\min\{\delta, \gamma\}$.

Based on the discussion so far, we can now present two situations (and a third situation a bit later) under which $V(k)$ is strictly decreasing. These situations will play an important role in the proof of the main result.

- **Situation 1 (S1)** occurs if at iteration k there exists a link in the network between a node $j \in \{X_4 \cup X_5 \cup X_6\}$ and a node $i \in \{X_1 \cup X_2\}$, in this case we have,

$$\nabla V_k \leq -\min\{\gamma, \delta\}, \tag{4.34}$$

  where $\delta = \min_{(i,j)\in E} w_{ij} > 0$.

- **Situation 2 (S2)** occurs if at iteration k there exists any link in the network between a node $j \in X_5 \cup X_6$ and a node $i \in X_3$, in this case we have,

$$\begin{aligned} \nabla V_k &\leq -\min\{\alpha_i - c_i(k), w_{ij}\} \\ &\leq -\min\{\gamma, \delta\}. \end{aligned} \tag{4.35}$$

4.5.5 Proof of Main Result

To show that $V(k)$ is eventually decreasing, we have to introduce some more notation. Let

$$R(k_0) = \min\{k - k_0; k > k_0, \nabla V_k \leq -\beta\},$$

where $\beta > 0$ is a positive constant. Notice that if either S1 or S2 occurs at time $T_0 > k_0$, then $R(k_0) \leq T_0 - k_0$ by considering $\beta = \min\{\gamma, \delta\}$, i.e., $R(k_0)$ is upper bounded by the minimum time for at least one of the two situations to occur. We will show that if there exists at least one node in $\{X_4, X_5, X_6\}$ at k_0 and $m(k) = m(k_0)$ for $k < R(k_0) + k_0$, then we can have a fixed upper bound on $R(k_0)$. If we looked at the values of the nodes in the network at any iteration k_0, we can see that if $k < k_0 + R(k_0)$, the network has a special structure: only nodes in $\{X_1, X_2, X_3\}$ have links between each other, nodes in X_3 can also have links to X_4, but not to $\{X_5, X_6\}$. Nodes in $\{X_5, X_6\}$ can only be connected to X_4 (see Fig. 4.4). Moreover, the values of nodes in X_3 cannot increase due to the link between X_3 and X_4. To see this, let $i \in X_3$ and $s \in X_4$ where $s \in N_i$. Then we have:

$$x_i(k+1) = x_i(k) + w_{is}L_{si} + \sum_{j\in N_i - \{s\}} w_{ij}L_{ji},$$

but since $\lfloor x_i(k) \rfloor = \lfloor x_s(k) \rfloor$, we have $L_{is} = 0$ and thus $x_i(k+1) = x_i(k) + \sum_{j\in N_i-\{s\}} w_{ij}L_{ji}$, so nodes in X_4 do not have any effect on nodes in X_3 and the values of nodes in X_3 cannot increase for all $k < k_0 + R(k_0)$ (we will get back to this issue later).

To find the number of iterations for a dotted (red) link to appear, we define the following function for nodes in $\{X_1, X_2, X_3\}$:

$$f(i,k) = \begin{cases} 1 & \text{if } i \in \{X_1(k), X_2(k)\}, \\ 0 & \text{if } i \in X_3(k), \end{cases} \tag{4.36}$$

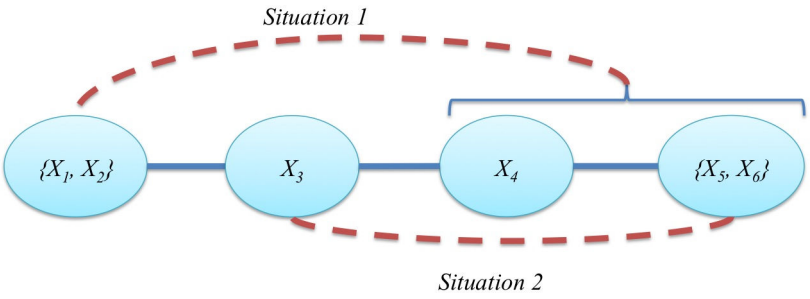

Figure 4.4: The solid lines (blue links) identify the network structure at any iteration $k_0 \leq k < k_0 + R(k_0)$, while if a dotted link (in red) appears, then $V(k)$ strictly decreases.

and let $T_i(k_0, k)$ be the number of times a node i is in $\{X_1, X_2\}$ in the time interval between k_0 and k, i.e.,

$$T_i(k_0, k) = \sum_{t=k_0}^{t=k} f(i, t).$$

In fact, we can partition the nodes in $\{X_1, X_2, X_3\}$ depending on their distance to nodes in X_4. Let r_i be the shortest path distance from a node $i \in \{X_1, X_2, X_3\}$ to the set X_4 (i.e., $r_i = \min_{j \in X_4} r_{ij}$ where r_{ij} is the number of hops following the shortest path from i to j). We define the set D_u where $u = 1, \ldots, r$ and $r = \max_i r_i$ as the set of nodes such that $i \in D_u$ if and only if $u = r_i$. For example, D_1 contains nodes that have direct neighbors in X_4, D_2 contains the nodes that do not have direct neighbors in X_4 but there is a node in X_4 found 2 hops away, and so on. Moreover, for any node $i \in D_u$ such that $u > 1$, we can find at least one neighbor $j \in D_{u-1}$. Let $P(i)$ be any one of these neighbors, referred to as the parent of i. It is important to note that any node in D_u remains in the set as long as non of the situation has occurred, i.e., the sets D_u for $u = 1, \ldots, r$ considered at iteration k_0 do not change their elements for $k_0 \leq k < k_0 + R(k_0)$. We can now obtain the following lemma:

Lemma 8. *If $\{X_4, X_5, X_6\} \neq \phi$ at an iteration k_0, and $m(k) = m(k_0)$ for $k_0 \leq k < k_0 + R(k_0)$, then for any integer $N \in \mathbb{N}$: if*

$$T_i(k_0, k) \geq N \times \left(\frac{\alpha_{P(i)}}{w_{iP(i)}} + 1 \right),$$

then

$$T_{P(i)}(k_0, k) \geq N.$$

Proof. The proof is based on the observation we mentioned earlier. For any node $s \in X_3$, its neighbors in X_4 do no have any effect on $x_s(k+1)$ and it cannot have any neighbor in $\{X_5, X_6\}$ otherwise one of the situations (S1 or S2) occurs and contradicts the assumption $k < k_0 + R(k_0)$. Therefore, the decrease of the node s

from X_3 to X_2 can only be due to its neighbors in $\{X_1, X_2\}$. Let $i \in \{X_1, X_2\}$ be a neighbor of node s, then

$$\begin{aligned} x_s(k+1) &= x_s(k) + \sum_{j \in N_s} w_{js} L_{js} \\ &= x_s(k) + w_{is} \times (-1) + \sum_{j \in N_s \cap \{X_1, X_2\} - \{i\}} w_{js} L_{js} \\ &\leq x_s(k) - w_{is} \\ &= 1 + m + c_s(k) - w_{is}, \end{aligned}$$

and the node s can either drop to X_2 or stay in X_3 depending on the resulting value $x_s(k+1)$. And since $c_s(k) \leq \alpha_s$ and $x_s(k+1)$ cannot increase if s was in X_3 at iteration k, then we are sure that if i was in $\{X_1, X_2\}$ for more than $\frac{\alpha_s}{w_{is}}$ iterations (i.e., $T_i(k_0, k) \geq \frac{\alpha_s}{w_{is}} + 1$), then s has dropped to X_2 at least once (i.e., $T_s(k_0, k) \geq 1$). Thus since $P(i) \in N_i$, we have

$$T_i(k_0, k) \geq \left(\frac{\alpha_{P(i)}}{w_{iP(i)}} + 1 \right) \quad \Longrightarrow \quad T_{P(i)}(k_0, k) \geq 1. \tag{4.37}$$

If $T_i(k_0, k_N) \geq N \times \left(\frac{\alpha_{P(i)}}{w_{iP(i)}} + 1 \right)$, then we can find $N - 1$ iterations, $k_1, k_2, \ldots, k_{N-1}$, such that

$$T_i(k_{v-1}, k_v - 1) \geq \left(\frac{\alpha_{P(i)}}{w_{iP(i)}} + 1 \right) \quad \text{for } v = 1, \ldots, N.$$

By (4.37), we have $T_{P(i)}(k_{v-1}, k_v - 1) \geq 1$. Therefore,

$$\begin{aligned} T_{P(i)}(k_0, k) &= \sum_{v=1}^{N-1} T_{P(i)}(k_{v-1}, k_v - 1) + T_{P(i)}(k_{N-1}, k) \\ &\geq \left(\sum_{v=1}^{N-1} 1 \right) + 1 \\ &\geq N, \end{aligned}$$

and the lemma is proved. □

Now we show that there is a fixed upper bound on the time for either of the situations to occur,

Lemma 9. *If $\{X_4, X_5, X_6\} \neq \phi$ at an iteration k_0, and $m(k) = m(k_0)$ for $k \geq k_0$, then*

$$R(k_0) \leq n \left(1 + \frac{1}{2\delta} \right)^{n-1},$$

where $\delta = \min_{(i,j) \in E} w_{ij}$ is a positive constant ($\delta > 0$).

Proof. Notice first that for any iteration $\bar{k} \geq k_0$, if $T_i(k_0, \bar{k}) \geq 1$ where $i \in D_1$, then situation 1 has occurred and $R(k_0) \leq \bar{k} - k_0$.

Moreover, since $m(k) = m(k_0)$ for $k \geq k_0$, then at every iteration k there is at least one node in $\{X_1, X_2\}$, leading to

$$\sum_{i \in \{X_1, X_2, X_3\}} T_i(k_0, k) \geq k - k_0.$$

Let $\bar{k} = k_0 + n\left(1 + \frac{1}{2\delta}\right)^{n-1}$; then we have

$$\sum_{i \in \{X_1, X_2, X_3\}} T_i(k_0, \bar{k}) \geq n\left(1 + \frac{1}{2\delta}\right)^{n-1},$$

and there must be a node $i \in D_u$ in this sum such that

$$T_i(k_0, \bar{k}) \geq \left(1 + \frac{1}{2\delta}\right)^{n-1}.$$

Without loss of generality, we can suppose $\frac{1}{2\delta} \in \mathbb{N}$. So applying Lemma 8, we can see that

$$\begin{aligned} T_i(k_0, \bar{k}) &\geq \left(1 + \frac{1}{2\delta}\right)^{n-1} \\ &\geq \left(1 + \frac{\alpha_{P(i)}}{w_{iP(i)}}\right) \times \left(1 + \frac{1}{2\delta}\right)^{n-2}, \\ &= \left(1 + \frac{\alpha_{P(i)}}{w_{iP(i)}}\right) \times N, \end{aligned}$$

where $N = \left(1 + \frac{1}{2\delta}\right)^{n-2}$, which implies

$$T_j(k_0, \bar{k}) \geq \left(1 + \frac{1}{2\delta}\right)^{n-2},$$

where $j = P(i)$ and $j \in D_{u-1}$. Doing this recursively ($u - 1$ times), we see that there is a node $s \in D_1$ such that,

$$T_s(k_0, \bar{k}) \geq \left(1 + \frac{1}{2\delta}\right)^{n-u},$$

but since $u \leq r \leq n$, we have $T_s(k_0, \bar{k}) \geq 1$ which means situation S1 occurred because $s \in D_1$. Therefore,

$$\begin{aligned} R(k_0) &\leq \bar{k} - k_0 \\ &\leq n\left(1 + \frac{1}{2\delta}\right)^{n-1}, \end{aligned}$$

and the lemma is proved. □

We also need the following lemma,

Lemma 10. *Suppose Assumption 1 holds. Let $\beta = \min\{\gamma, \delta\}$, then for the quantized system (4.12), at any time k_0, there is a finite time $k_1 \geq k_0$ such that for $k \geq k_1$, either $\{X_4, X_5, X_6\} = \phi$ or $m(k) > m(k_0)$. Moreover,*

$$k_1 \leq k_0 + n\left(\frac{V(k_0)}{\beta} + 1\right)\left(\frac{1}{2\delta} + 1\right)^{n-1}.$$

Proof. Let us prove it by contradiction. Suppose that $\{X_4, X_5, X_6\} \neq \phi$ and $m(k) = m(k_0)$ for $k \geq k_0$. Therefore we can apply Lemma 9 to show that there is an upper bound $R(k_0)$ for situations S1 or S2 to occur. Whenever one of the situations occurs, we have $\nabla V_k \leq -\beta$, otherwise $\nabla V_k \leq 0$. For $k > k_0 + n\left(\frac{V(k_0)}{\beta} + 1\right)\left(\frac{1}{2\delta} + 1\right)^{n-1}$, we have that situations S1 or S2 have occurred at least $\left(\frac{V(k_0)}{\beta} + 1\right)$ times; then

$$V(k) \leq V(k_0) - \beta \times \left(\frac{V(k_0)}{\beta} + 1\right) \leq -\beta < 0,$$

which is a contradiction since $V(k) \geq 0$ is a Lyapunov function. As a result, there exists an iteration k_1 satisfying $k_1 \leq k_0 + n\left(\frac{V(k_0)}{\beta} + 1\right)\left(\frac{1}{2\delta} + 1\right)^{n-1}$ such that for $k \geq k_1$, either $\{X_4, X_5, X_6\} = \phi$ or $m(k) > m(k_0)$. □

We are now ready to prove the following propositions,

Proposition 11. *Consider the quantized system (4.12). Suppose that Assumption 1 holds. Then for any initial value $\mathbf{x}(0)$, there is a finite time iteration where $\{X_4, X_5, X_6\} = \phi$.*

Proof. The value $m(k)$ cannot increase more than $M(0) - m(0)$ number of times because $M(k)$ is non-increasing. Therefore, applying Lemma 10 for $M(0) - m(0)$ times, we see that $\{X_4, X_5, X_6\} = \phi$ in a finite number of iterations. □

Proposition 11 shows that in fact the nodes are restricted in a finite number of iterations to the sets $\{X_1, X_2, X_3\}$. In fact, we can even show a stronger result, that either X_1 or X_3 can be nonempty, but not both. This is given in the next proposition.

Proposition 12. *Consider the quantized system (4.12). Suppose that Assumption 1 holds. Then for any initial value $\mathbf{x}(0)$, there is a finite time iteration where either $\{X_3, X_4, X_5, X_6\} = \phi$ or $\{X_1, X_4, X_5, X_6\} = \phi$.*

Proof. Due to Proposition 11, we can find a finite time T such that $\{X_4, X_5, X_6\} = \phi$. Without loss of generality, we consider $T = 0$. In fact, a third situation that can strictly decrease $V(k)$ occurs when there is a link between a node in X_1 and a node in X_3. Fig. 4.5 shows the network structure. If Situation 3 (S3) occurs and $(ij) \in E$

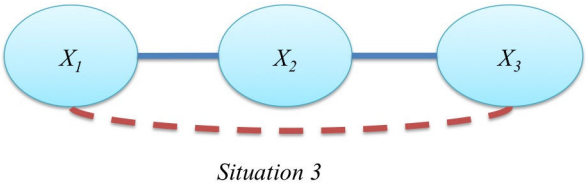

Figure 4.5: The solid lines (blue links) identify the network structure at any iteration $k_0 \leq k < k_0 + R(k_0)$, while if the dotted link (in red) appears, then $V(k)$ strictly decreases.

where $i \in X_1$ and $j \in X_3$, then

$$\begin{aligned} \nabla V_k &\leq -\min\{\bar{c}_i(k) - \alpha_i, w_{ij}\} \\ &\leq -\min\{\gamma, \delta\}. \end{aligned} \tag{4.38}$$

In fact, similar to the reasoning along this subsection, we can bound the number of iterations for S3 to occur. The bound is exactly the same as the one developed for the other situations. Instead of repeating the derivations, the proof reads roughly the same starting from the beginning of Subsection 4.5.5 but by replacing X_1, X_2, and X_3 by ϕ, replacing X_2 by X_3, replacing X_3 by X_2, replacing X_4 by X_1, and finally replacing the condition $m(k) = m(k_0)$ by $X_3 \neq \phi$. Thus, Lemma 10 will read as follows: Suppose Assumption 1 holds. Let $\beta = \min\{\gamma, \delta\}$, then for the quantized system (4.12), at any time k_0, there is a finite time $k_1 \geq k_0$ such that for $k \geq k_1$, either $X_1 = \phi$ or $X_3 = \phi$. This ends the proof. □

Proposition 13. *Consider the quantized system (4.12). Suppose that Assumption 1 holds and let $\alpha = \max_i \alpha_i$. Then for any initial value $\mathbf{x}(0)$, there is a finite time iteration where either*

- *the values of nodes are cycling in a small neighborhood around the average such that :*

$$\begin{cases} |x_i(k) - x_j(k)| \leq \alpha_i + \alpha_j \text{ for all } i, j \in V \\ |x_i(k) - x_{ave}| \leq 2\alpha \text{ for all } i \in V, \end{cases} \tag{4.39}$$

- *or the quantized values have reached consensus, i.e.,*

$$\begin{cases} \lfloor x_i(k) \rfloor = \lfloor x_j(k) \rfloor \text{ for all } i, j \in V \\ |x_i(k) - x_{ave}| < 1 \text{ for all } i \in V. \end{cases} \tag{4.40}$$

Proof. The two possibilities are consequence of the two possible cases of Proposition 12,

- Case $\{X_1, X_4, X_5, X_6\} = \phi$. Then all nodes are in $\{X_2, X_3\}$ and by the definition of the sets we have $|x_i(k) - x_j(k)| \leq \alpha_i + \alpha_j$ for all $i, j \in V$, so nodes are

cycling (due to Proposition 10) around $m+1$. Moreover, since the average is conserved from Eq. (4.9), we have:

$$\begin{aligned} |x_i(k) - x_{ave}| &= |x_i(k) - x_{ave}(k)| \\ &\leq |\max_i x_i(k) - \min_i x_i(k)| \\ &\leq 2 \max_i \alpha_i \\ &= 2\alpha, \end{aligned}$$

- Case $\{X_3, X_4, X_5, X_6\} = \phi$. Then all nodes are in $\{X_1, X_2\}$ and by the definition of the sets we have reached quantized consensus. Since for any i and j we have $c_i(k), c_j(k) \in [0, 1)$, then $|x_i(k) - x_j(k)| < 1$ and as in the above due to Eq. (4.9), we have $|x_i(k) - x_{ave}| < 1$.

□

4.6 Discussion

Propositions 10 shows that the uniform quantization on communications given by the model of this chapter can have a very important cyclic property. Up to our knowledge, this is the first work in deterministic quantized algorithms that shows this cyclic effect of nodes' values and it is also shown by Proposition 13 that the cyclic values can be control by a simple distributed adjustment of the weights. This can have an important impact on the design of quantized communication algorithms.[4] For example, due to the cyclic effect, nodes can use the history of their values to reach asymptotic convergence as the following proposition shows:

Corollary 1. *Consider the quantized system (4.12). Suppose that Assumption 1 holds. Then for any initial value* $\mathbf{x}(0)$*, if* $y_i(k)$ *is an estimate of the average at node* i *following the recursion:*

$$y_i(k) = \frac{k}{k+1} y_i(k-1) + \frac{1}{k+1} x_i(k), \quad \forall i \in V, \tag{4.41}$$

where $y_i(0) = x_i(0)$*, then* $y_i(k)$ *is converging,*

$$\lim_{k\to\infty} y_i(k) = y_i^*, \quad \forall i \in V, \tag{4.42}$$

having

$$|y_i^* - x_{ave}| \leq 1.$$

[4] Pattern generation (as for cyclic systems) plays an important role in the design of many mechanical and electrical systems [Bro97].

Proof. The state equation of $y_i(k)$ for a node i is give by

$$y_i(k) = \frac{k}{k+1} y_i(k-1) + \frac{1}{k+1} x_i(k) = \frac{1}{k+1} \sum_{t=0}^{t=k} x_i(t)$$
$$= \frac{1}{k+1} \left(\sum_{t=0}^{t=T_{conv}-1} x_i(t) \right) + \frac{1}{k+1} \left(\sum_{t=T_{conv}}^{t=k} x_i(t) \right),$$

where T_{conv} is the finite time iteration when the nodes' values start cycling. As k approaches infinity, the left part in the sum vanishes while the right part converges to the average of the values in a cycle, i.e.,

$$\lim_{k\to\infty} y_i(k) = y_i^* = \frac{1}{P} \sum_{t=T_{conv}}^{t=T_{conv}+P-1} x_i(t),$$

where P is the cycle period. Since for $k \geq T_{conv}$ we have $|x_i(k) - x_{ave}| \leq 1$ from Proposition 13, then $|y_i^* - x_{ave}| \leq 1$. □

Moreover, since the final behavior of the system depends on the initial values as shown by Proposition 13, we give here a condition on the initial values for the nodes to reach quantized consensus in networks:

Corollary 2. *Consider the quantized system* (4.12). *Suppose that Assumption 1 holds. If the initial values* $\mathbf{x}(0)$ *satisfy,*

$$\alpha \leq x_{ave} - \lfloor x_{ave} \rfloor \leq 1 - \alpha, \tag{4.43}$$

then the network reaches quantized consensus.

Proof. If the system was cyclic, then for any node $i \in V$, we have $i \in \{X_1, X_2\}$, so $x_i(k) \in [m + 1 - \alpha_i, m + 1 + \alpha_i]$. This implies that $x_{ave}(k) \in [m + 1 - \alpha_i, m + 1 + \alpha_i]$, but since the average is conserved (from equation (4.9)), it also implies that $x_{ave} \in [m + 1 - \alpha_i, m + 1 + \alpha_i]$. From the latter condition, we see that if $\alpha < x_{ave} - \lfloor x_{ave} \rfloor < 1 - \alpha$, the system cannot be cyclic, and by Proposition 13, it must reach quantized consensus. □

4.6.1 Design of Weights with Arbitrarily Small Error

If the system has reached quantized consensus, the values of the agents' agreement variables become stationary and the deviation of these values from the average is no larger than 1. In the case when the system does not reach quantized consensus but becomes cyclic, Proposition 13 shows that the deviation of nodes' values from the average is upper bounded by 2α where $\alpha = \max_i \alpha_i$. Moreover the deviation can be made arbitrarily small by adjusting the weights in a distributed manner. Toward that end, we propose the following modified Metropolis weights:

$$w_{ij} = \frac{1}{C\left(\max\{d_i, d_j\} + 1\right)}, \quad \forall (i,j) \in E$$
$$w_{ii} = 1 - \sum_{j \in N_i} w_{ij}, \quad \forall i \in V$$

where C is any rational constant such that $C \geq 2$. It can be easily checked that the proposed weights satisfy Assumption 1. Moreover, in addition to its distributed nature, the choice of C can be used to define the error. Notice that for any $i \in V$, we have

$$\begin{aligned} w_{ii} &= 1 - \sum_{j \in N_i} \frac{1}{C\left(\max\{d_i, d_j\} + 1\right)} \\ &\geq 1 - \frac{1}{C} \sum_{j \in N_i} \frac{1}{d_i + 1} \\ &= 1 - \frac{1}{C} \frac{d_i}{d_i + 1} \\ &= 1 - \frac{1}{C} + \frac{1}{C(d_i + 1)}, \end{aligned}$$

then $1 - w_{ii} \leq \frac{1}{C} - \frac{1}{C(d_i+1)}$, so

$$\begin{aligned} \alpha_i &= 1 - w_{ii} + \gamma \\ &\leq \frac{1}{C} - \frac{1}{C(d_i + 1)} + \gamma \end{aligned}$$

Since γ can be chosen arbitrarily from the interval $(0, \gamma_{\max}]$, by considering a small enough γ the following holds

$$\alpha \leq \frac{1}{C}.$$

This shows that given an arbitrary level of precision known to all the agents, the agents can choose the weights with large enough C in a distributed manner, so that the neighborhood of the cycle will be close to the average with the given precision. Notice that if $x_{ave} \neq \lfloor x_{ave} \rfloor$, then for α small enough, the system cannot be cyclic and only quantized consensus can be reached (Corollary 2). In other words, for systems starting with different initial values, having a smaller α leads more of these systems to converge to quantized consensus (and of course if they cycled, they will cycle in a smaller neighborhood as well due to Proposition 13).

It is worth mentioning that this arbitrarily small neighborhood weight design has a trade-off with the speed of convergence of quantized consensus protocol (small error weight design leads to slower convergence).

4.7 Simulations

In this section, we present some simulations to demonstrate the theoretical results in the previous section. The weights for the simulations satisfy Assumption 1 and are the modified Metropolis weights with $C = 2$, i.e.,

$$w_{ij} = \frac{1}{2\left(\max\{d_i, d_j\} + 1\right)} \quad \forall (i, j) \in E.$$

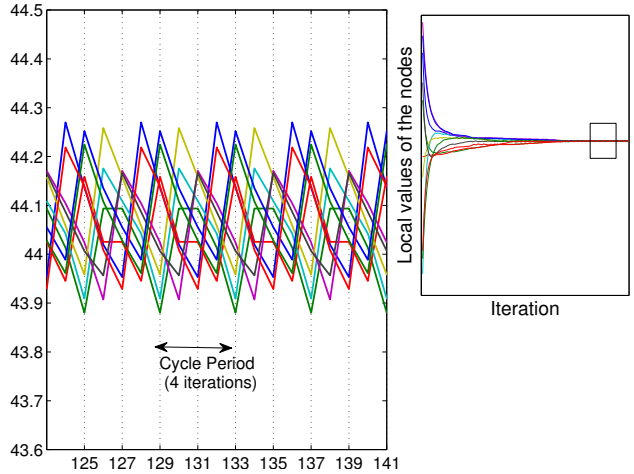

Figure 4.6: The nodes' values are entering into a cycle.

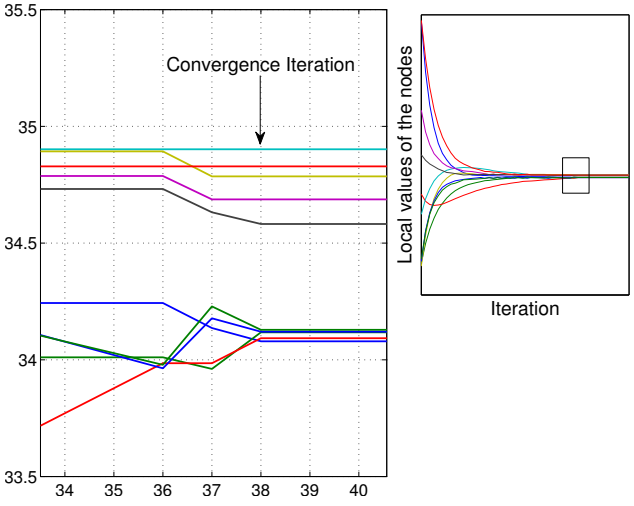

Figure 4.7: The nodes' values are converging.

4.7.1 Simple Network

Proposition 13 shows that depending on the initial state $\mathbf{x}(0)$, the system reaches in finite time one of the two possibilities: 1) cyclic, 2)quantized consensus. We show on a network of 10 nodes with initial values selected uniformly at random from the interval $[0, 100]$ that both of these are possible. Fig. 4.6, shows that after a certain iteration, the nodes' values enter into a cycle of period 4 iterations, while Fig. 4.7 shows that starting from different initial values, all the 10 nodes reach quantized consensus in finite time. Mainly, at iteration 38, all nodes' values are between 34 and 35; therefore, we have

$$\lfloor x_i(k) \rfloor = 34 \qquad \forall i = 1, \ldots, 10, \;\; \forall k \geq 38.$$

4.7.2 Random Graphs

To further simulate our theoretical results, we need to select some network model. The simulations are done on random graphs: Erdös-Renyi (ER) graphs and Random Geometric Graphs (RGG), given that they are connected. The random graphs are generated as follows:

- For the ER random graphs, we start from n nodes fully connected graph, and then every link is removed from the graph by a probability $1 - P$ and is left there with a probability P. We have tested the performance for different probabilities P given that the graph is connected.

- For the RGG random graphs, n nodes are thrown uniformly at random on a unit square area, and any two nodes within a connectivity radius R are connected by a link (the connectivity radius R is selected as $R = \sqrt{c \times \frac{\log(n)}{n}}$ where c is a constant that is studied by wide literature on RGG for connectivity). We have tested the performance for different connectivity radii given that the graph is connected. It is known that for a small connectivity radius, the nodes tend to form clusters.

Since Proposition 13 shows that the system would reach one of the cases in finite time, let us define T_{conv} be this time. Notice that if nodes enter the cyclic states (case 1), the Lyapunov function is null because for all $i \in V$ and $k \geq T_{conv}$, we have $x_i(k) \in [m + 1 - \alpha_i, m + 1 + \alpha_i]$, so we can write,

$$V(k) = 0 \;\; \forall k \geq T_{conv}.$$

However, if nodes reached quantized convergence (case 2), then the Lyapunov function is a constant because for all $i \in V$ and $k \geq T_{conv}$, we have $x_i(k) \in [m, m + 1]$, so we can write,

$$V(k) = cte \;\; \forall k \geq T_{conv}.$$

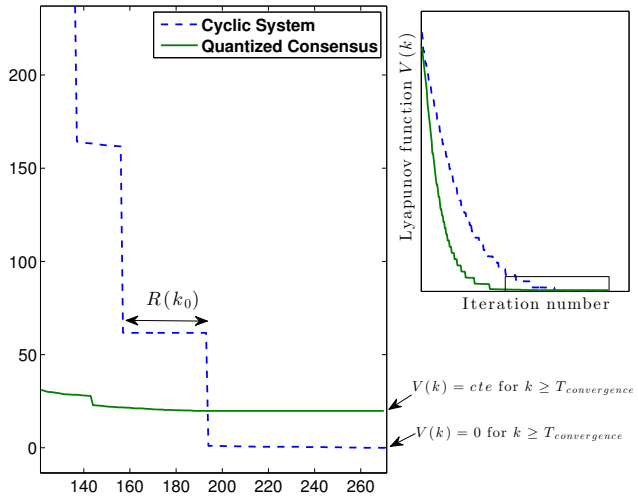

Figure 4.8: The system Lyapunov function $V(k)$.

4.7.2.1 Lyapunov Function

Fig. 4.8 shows the Lyapunov functions for the two different cases on an RGG with 100 nodes and $R = 0.2146$, where each case corresponds to initial values of nodes selected uniformly at random from the interval $[0, 100]$. The figure also shows $R(k_0)$ which is the number of iterations after k_0 up till $V(k)$ decreases (S1 or S2 occurs).

4.7.2.2 Quantized Consensus

Given that we are considering Metropolis weights with $C = 2$, then the system satisfies (4.43) if initial states are such that $x_{ave} - \lfloor x_{ave} \rfloor = 0.5$. We considered RGG and ER graphs of 100 nodes, where the initial condition is chosen as follows: the first 99 nodes are given uniformly random initial values from the interval $[0, 100]$, while the last node is given an initial value such that $x_{ave} - \lfloor x_{ave} \rfloor = 0.5$ is satisfied. Therefore, with these initial values, by applying Corollary 2, the system reaches quantized consensus in finite time T_{conv}. Table I shows the mean value over 100 runs of the T_{conv} for the RGG with different connectivity radii, $R_1 < R_2 < R_3 < R_4 < R_5$, where $R \in \{0.1357, 0.1517, 0.1858, 0.2146, 0.3717\}$. The results show that the more the graph is connected, the faster the convergence. These results are also shown to be true on ER graphs. Table II shows the mean value over 100 runs of the T_{conv} for the ER with different probability P , $P_1 < P_2 < P_3 < P_4$, where $P \in \{0.04, 0.06, 0.08, 0.10\}$.

4.8 Conclusion

In this chapter, we studied the performance of deterministic distributed averaging protocols subject to communication quantization. We have shown that depending

	RGG $n = 100$				
	R_1	R_2	R_3	R_4	R_5
T_{conv}	1965.3	1068.9	364.3	233.3	55.9

Table 4.1: Convergence time for Random Geometric Graphs (RGG) with different connectivity radii (averaged over 100 runs).

	ER $n = 100$			
	$P_1 = 0.04$	$P_2 = 0.06$	$P_3 = 0.08$	$P_4 = 0.10$
T_{conv}	161.49	99.38	66.58	43.43

Table 4.2: Convergence time for Erdos Renyi (ER) with different probabilities of link existence (averaged over 100 runs).

on initial conditions, the system converges in finite time to either a quantized consensus, or the nodes' values are entering into a cyclic behavior oscillating in a small neighborhood around the average. The size of this neghborhood can be controlled by a decentralized design of the weight matrix. We also provided conditions for which quantized consensus is guaranteed.

Chapter 5

Reducing Communication Overhead

Contents

As demonstrated in the Introduction, the asymptotic convergence rate of consensus protocols depends on the selected weights. In Chapter 1 we have proposed an optimization problem that selects the weights in consensus protocols to achieve fast asymptotic convergence rates. However, speeding up this rate does not automatically reduce the number of messages that are sent in the network. The reason is that the convergence is reached only asymptotically, and even if nodes' estimates are very close to the average, nodes keep on performing the averaging and sending messages to their neighbors.

In this chapter, we address this issue. We propose an algorithm that relies only on limited local information to reduce communication overhead for average consensus. As the nodes' estimates approach the true average, nodes exchange messages with their neighbors less frequently. The algorithm has a nice self-adaptive feature: even if it has already converged to a stable state and the message exchange rate is very small, when an exogenous event leads the value at a node to change significantly, the algorithm detects the change and ramps up its communication rate. The proposed algorithm provides also a trade-off between the precision of the estimated average and the number of messages sent in the network by setting one of its parameter. Being totally decentralized, the message reduction algorithm can also be applied in a dynamic network with faulty links.

5.1 System equation

The system equation of this chapter at iteration $k+1$, node i updates its state value x_i:[1]

$$x_i(k+1) = w_{ii}x_i(k) + \sum_{j \in N_i} w_{ij}x_j(k). \tag{5.1}$$

For a node i to have access to the values of its neighbors' variables, each node $j \in N_i$ should send the value $x_j(k)$ to i before the iteration $k+1$ takes place. The communication overhead due to these messages can be a burden on the network if the algorithm ran for a long time.

The matrix form equation is:

$$\mathbf{x}(k+1) = W\mathbf{x}(k). \tag{5.2}$$

In this chapter, we consider W to be $n \times n$ real doubly stochastic matrix having $\mu(W) < 1$ where $\mu(W)$ is the second largest eigenvalue in module of W. We also consider that W is constructed locally (e.g., using the Metropolis weights described in Chapter 4). With these conditions on W, the convergence to the average consensus is in general asymptotic:

$$\lim_{k\to\infty} \mathbf{x}(k) = x_{ave}\mathbf{1}. \tag{5.3}$$

Since average consensus is usually reached only asymptotically in (5.3), the nodes will always be busy sending messages. Let $N(k)$ be the number of nodes transmitting at iteration k, so without a termination procedure all nodes are transmitting at iteration k, $N(k) = n$ independently from the current estimates. In this chapter we present an algorithm that reduces communication overhead and provides a trade-off between precision of the consensus and number of messages sent.

5.2 Related Work

Some previous works considered protocols for average consensus protocol to terminate (in finite time) to converge to the exact average or to guaranteed error bounds. For example, the approach proposed in [SH07] is based on the *minimal polynomial* of the matrix W. The authors show that a node, by using coefficients of this polynomial, can calculate the exact average from its own estimate on K consecutive iterations. The drawback is that nodes must have high memory capabilities to store $n \times n$ matrix, and high processing capabilities to calculate the coefficients of the minimal polynomial by solving a set of n linearly independent equations. Another approach for finite time termination is given in [YS07], where the proposed algorithm does not calculate the exact average, but estimates are guaranteed to be within a predefined distance from the average. This approach runs three consensus

[1] At one point in the simulations in this chapter, the topology of the network may change dynamically. This is taken into account in (5.1) by letting the neighborhood and the weights be time-dependent (then we have $N_i(k)$ and $w_{ij}(k)$).

protocols at the same time: the average consensus which runs continuously and the maximum and the minimum consensus restarted every U iterations where U is an upper bound on the diameter of the network. The difference between the maximum and the minimum consensus provides a stopping criteria for nodes.

Under the assumption of asynchronous iterations, the authors in [DRL11] proposed an algorithm that leads to the termination of average consensus in finite time with high probability. In their approach, each node has a counter c_i that stores the number of times the difference between the new estimate and the old one was less than a certain threshold τ. When the counter reaches a certain value, say C, the node will stop initiating the algorithm. They proved that by a correct choice of C and τ (depending on some networks' parameters as the maximum degree in the network, the number of nodes, and the number of edges) the protocol terminates with high probability.

A major drawback of these algorithms –beside the memory requirements and the robustness of the system to changes– is the assumption that each node should know some global network parameters. This intrinsically contradicts the spirit of distributed consensus protocols. Designing a decentralized algorithm for average consensus that terminates in finite time without using any global network information (as the diameter of the network or the number of nodes) is still an open problem for which we prove a strong negative result in the next section.

5.3 Motivation

We address the problem of termination of average consensus in this chapter. We will start by an impossibility result for termination of the average consensus protocol in finite time without using some network information.

Theorem 2. *Given a static network where nodes run the synchronous consensus protocol described by* (5.1) *and each node only knows its history of estimates, there is no deterministic distributed algorithm that can correctly terminate the consensus with guaranteed error bounds after a finite number of steps for any set of initial values.*

Proof. The proof is conducted by contradiction where we show that there exists a graph with specific initial state values which fails to terminate with guaranteed error bounds. Consider a path graph G of three nodes a, b, and c as in Fig. 5.1 where the weight matrix is real and doubly stochastic with $0 \leq \mu(W) < 1$ (so we have $w_{aa}, w_{cc} > 0$). Let $x_a(0)$, $x_b(0)$, and $x_c(0)$ be the initial estimates for the nodes and consider $\alpha = \frac{x_a(0)+x_b(0)+x_c(0)}{3}$, so with the average consensus protocol using the synchronous iterations in (5.1), all nodes' estimates will converge to α asymptotically:

$$\lim_{k\to\infty} x_a(k) = \lim_{k\to\infty} x_b(k) = \lim_{k\to\infty} x_c(k) = \alpha.$$

We will prove the theorem by contradiction. Suppose there exists a termination algorithm for nodes to use only the history of their estimates and terminate the

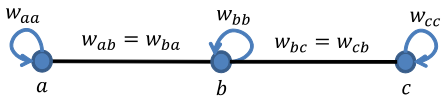

Figure 5.1: Path graph G with 3 nodes.

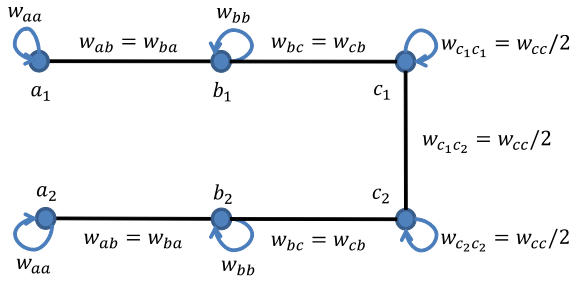

Figure 5.2: Extended mirror graph of G with 6 nodes and $F = 2$ fragments.

average protocol in finite time within guaranteed error bounds. Then if we run this algorithm on this graph, there exists an iteration $K > 0$ and $\eta > 0$ such that node a (also true for b and c) decides to terminate at iteration K on the basis of the history of its estimate: $x_a(0), x_a(1), x_a(2), ..., x_a(K)$, and it is guaranteed that $|x_a(K) - x_{ave}| < \eta$, where $x_{ave} = \alpha$.

We will define the F extended mirror graph of G to be a path with $n = 3F$ nodes $a_1, a_2, ..., a_F, b_1, b_2, ..., b_F, c_1, c_2, ..., c_F$, formed by $G_1, G_2, ..., G_F$ (F graphs identical to G) connected by additional links to form a path, the added links are $\{c_l, c_{l+1}\}$ if l is odd and $\{a_l, a_{l+1}\}$ if l is even (e.g. the graph for $F = 2$ is shown in Fig. 5.2). Let us assume first that the initial estimates for nodes in the subgraphs G_1,...,G_F are identical to the estimates of the nodes in graph G (e.g. for node a we have $x_{a_1}(0) = x_{a_2}(0) = ... = x_{a_F}(0) = x_a(0)$), the weight matrix for G_1,...,G_F is also identical to the weight matrix of G except for nodes incident to the added links, if $\{c_l, c_{l+1}\}$ is an added link, then $w_{c_lc_l} = w_{c_{l+1}c_{l+1}} = w_{c_lc_{l+1}} = \frac{w_{cc}}{2}$ and similarly if $\{a_l, a_{l+1}\}$ is an added link, then $w_{a_la_l} = w_{a_{l+1}a_{l+1}} = w_{a_la_{l+1}} = \frac{w_{aa}}{2}$. Notice that on the new generated graph we still have $x_{ave} = \alpha$ and also:

$$x_{a_1}(k) = x_{a_2}(k) = ... = x_{a_F}(k) = x_a(k) \ \forall k \leq K,$$

so node a_1 applying the termination algorithm on the new graph will decide to terminate after the same number of iterations K. Consider now a value $F > K$ and that the initial estimate of node c_{K+1} is changed to $x_{c_{K+1}}(0) = x_c(0) + n(2\eta + x_{a_1}(K) - \alpha)$ and the new average is now $x_{ave} = \alpha + \frac{x_{c_{K+1}}(0) - x_c(0)}{n}$. The estimates at node a_1 would not change during the first K steps, then node a_1 would again terminate at step K, but the error bound is no more guaranteed, because $|x_{a_1}(K) - x_{ave}| = |x_{a_1}(K) - \alpha - \frac{x_{c_{K+1}}(0) - x_c(0)}{n}| = 2\eta > \eta$. This contradicts the fact that a_1 terminates with **guaranteed** error bounds. The proof can be extended to include

any graph G, not just path graphs, by using the same technique of generating extended mirror graphs of G. □

Theorem 1 shows that in general, nodes cannot stop executing the algorithm. Motivated by this result, we investigate in what follows algorithms where nodes can refrain from sending messages at every iteration (e.g. when estimates have not changed significantly during the recent iterations). We will then say that an algorithm terminates when the number of messages sent in the network disappears at least asymptotically, even if the nodes are still running the algorithm internally, i.e.,

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{t} N(k)}{t} = 0, \tag{5.4}$$

where $N(k)$ is the number of nodes transmitting their estimate to their neighbors at iteration k. In other words, the rate of messages in the network should decrease as the estimates converge to the average consensus or to a bounded approximation.

5.4 Our Approach

Even if the nodes cannot terminate the algorithm in finite time, we are interested in reducing communication overhead by considering asymptotic termination of messages and by decreasing the rate of the messages sent in the network correspondingly to estimates' improvement. For example, if nodes' estimates are widely different, the messages sent at a given iteration can significantly reduce the error by making the estimates approach to the real average. However, when the estimates have "almost converged", the improvement from each message in terms of error reduction can be negligible. Up to our knowledge, this issue was not taken into account in the related work literature. So from an engineering perspective, it is desirable that nodes send more messages when they have large differences in their estimates, and less messages when the estimates have almost converged. In what follows we first present a centralized algorithm to provide the intuition of our approach and then we describe a more practical decentralized solution.

5.4.1 A Centralized Algorithm

In this section, we discuss a simple centralized algorithm for termination of average consensus protocols. We call it a centralized protocol because in this protocol there are some global variables known to all the nodes in the network, and each node can send a broadcast signal that triggers an averaging operation (5.1) at all nodes. Then, if any of the nodes in the network sends this signal, all the nodes will respond by sending the new estimates to their neighbors according to the averaging equation (5.2):

$$\mathbf{x}(t+1) = W\mathbf{x}(t). \tag{5.5}$$

On the contrary, if no signal is sent, the nodes will preserve the same estimate:

$$\mathbf{x}(t+1) = \mathbf{x}(t). \tag{5.6}$$

If the rate of broadcast signals converges to 0, also the rate of the messages containing the estimates will converge to 0 and asymptotically no node in the network will transmit. As above we consider a time-slotted model where t represents a discrete time iteration.

We now introduce formally the algorithm. Let $e(t)$ and $\eta(t)$ be the values of two global variables known to all the nodes at time t, such that $e(0) = 0$, $\eta(0) = \eta_0$ and $0 \leq e(t) < \eta(t)$. As we are going to see, both the values of the two variables cannot decrease. Let W be the weight matrix of the network satisfying convergence conditions of average consensus and $\mathbf{x}(t)$ be the state vector of the system at iteration t. We let L_t be a Boolean variable (either true or false) defined at every iteration t as:

$$L_t : \ e(t-1) + y(t-1) < \eta(t-1), \tag{5.7}$$

where $y(t-1) = ||W\mathbf{x}(t-1) - \mathbf{x}(t-1)||_\infty$ and with $L_0 := False$. Then $y(t-1)$ stores the estimates change if the linear iterations (5.1) would be executed at step t and L_t evaluates if the change is negligible ($L_t = False$) and then no message is transmitted or not ($L_t = True$). Different actions are taken on the basis of the L_t value at timeslot t. We also define the simple point process $\psi = \{t_k : k \geq 1\}$ to be the sequence of strictly increasing points

$$0 < t_1 < t_2 < \dots ,$$

such that $\hat{t} \in \psi$ if and only if $L_{\hat{t}} = False$. Let $K(t)$ denote the number of points of the set ψ that falls in the interval $]0, t]$, i.e., $K(t) = \max\{k \ : \ t_k \leq t\}$, with $K(0) := 0$. If L_t is false, a broadcast signal is sent in the network and all nodes will perform an averaging iteration; while if L_t is true, then there is no signal in the network, and the nodes keep the same estimate as the previous iteration. Network variables of the centralized algorithm are changed at time $t > 0$ according to the equations given in following table:

If L_t is $True$	If L_t is $False$
$K(t) = K(t-1)$	$K(t) = K(t-1) + 1$
$\mathbf{x}(t) = \mathbf{x}(t-1)$	$\mathbf{x}(t) = W\mathbf{x}(t-1)$
$e(t) = e(t-1) + y(t-1)$	$e(t) = e(t-1)$
$\eta(t) = \eta(t-1)$	$\eta(t) = \eta(t-1) + \frac{\eta_0}{(K(t-1)+1)^2}$

When $t \notin \psi$, we call t a silent iteration because the nodes have the same estimate as the previous iteration (i.e., $x_i(t) = x_i(t-1)$) and there is no need to exchange messages of these estimates in the network. On the other hand, when $t \in \psi$, we call t as a busy iteration because nodes will perform an averaging (i.e., $\mathbf{x}(t) = W\mathbf{x}(t-1)$) and the estimates must be exchanged in the network. Let α_k be the number of silent iterations between t_k and t_{k+1}, so we have that $\alpha_k = t_{k+1} - t_k - 1$.[2]

After introducing this deterministic procedure, we show by the following lemma that the messages according to this algorithm disappear asymptotically:

[2] $t_k - t_{k-1}$ is sometimes called the k^{th} interarrival time in the context of point processes.

Proposition 14. *For any initial condition* $\mathbf{x}(0)$*, the message rate of the centralized deterministic algorithm described above disappears asymptotically, i.e.,*

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{t} N(k)}{t} = 0,$$

where $N(k)$ *is the number of nodes transmitting messages at iteration* k*.*

Proof. The number of nodes transmitting at an iteration t depends on the condition L_t. If $t \in \psi$, then $N(t) = n$ (all nodes are transmitting messages), otherwise $N(t) = 0$ (no nodes transmitting messages). Therefore,

$$\sum_{k=1}^{t} N(k) = \sum_{k=1}^{K(t)} N(t_k) = nK(t),$$

where $K(t)$ as described earlier is the number of busy iterations until time t. We will consider two cases depending on the evolution of $K(t)$ as function of t. The simpler case is when $\lim_{t\to\infty} K(t) \leq K < \infty$ (the number of busy periods is bounded, e.g. nodes reach consensus in a finite number of iterations), then since $K(t)$ is an increasing positive integer sequence, the proposition follows from the following inequality and $t \to \infty$,

$$0 \leq \frac{\sum_{k=1}^{t} N(k)}{t} \leq \frac{nK}{t}.$$

We consider now the other case, i.e., $\lim_{t\to\infty} K(t) = \infty$. Notice that for any time iteration t, we have

$$\sum_{k=1}^{K(t)} (t_k - t_{k-1}) \leq t \leq \sum_{k=1}^{K(t)+1} (t_k - t_{k-1}),$$

or in other words

$$\sum_{k=0}^{K(t)-1} (\alpha_k + 1) \leq t \leq \sum_{k=0}^{K(t)} (\alpha_k + 1).$$

So we have

$$\frac{\sum_{k=1}^{t} N(k)}{t} = \frac{nK(t)}{t} \leq \frac{nK(t)}{\alpha_{K(t)-1} + 1}$$

We will prove now that the right hand side of the inequality goes to 0 as t diverges. Since $\lim_{t\to\infty} K(t) = \infty$, it is sufficient to prove that $\lim_{k\to\infty}(\alpha_k + 1)/k = \infty$. Let $\mathbf{z}(k) = W\mathbf{x}(t_k) - \mathbf{x}(t_k)$, we can see that according to this algorithm,

$$\begin{aligned}
\alpha_k &= \lfloor \frac{\eta(t_k) - e(t_k)}{||\mathbf{z}(k)||_\infty} \rfloor \\
&\geq \frac{\eta(t_k - 1) + \eta_0/k^2 - e(t_k - 1)}{||\mathbf{z}(k)||_\infty} - 1 \\
&\geq \frac{\eta_0}{k^2||\mathbf{z}(k)||_2} - 1. \qquad (5.8)
\end{aligned}$$

The last inequality derives from the fact that for any iteration t we have $\eta(t) > e(t)$, and that for any vector $\mathbf{v}$, the norm inequality $||\mathbf{v}||_2 \geq ||\mathbf{v}||_\infty$ holds. Moreover, $\mathbf{z}(k)$ evolves according to the following equation:

$$\begin{aligned}\mathbf{z}(k) &= (W - J)\mathbf{z}(k-1) \\ &= (W - J)^k \mathbf{z}(0),\end{aligned}$$

where $J = 1/n\mathbf{1}\mathbf{1}^T$, so

$$||\mathbf{z}(k)||_2 \leq C\left(\rho(W - J)\right)^k, \tag{5.9}$$

where $C = ||\mathbf{z}(0)||_2$ and $\rho(W - J) = \mu(W) \geq 0$ is the spectral radius of the matrix $W - J$. We know that $0 < \mu < 1$ ($0 < \mu$ because $\lim_{t\to\infty} K(t) = \infty$ and $\mu < 1$ because W satisfies the condition of a converging matrix. Putting everything together, we get finally that:

$$\alpha_k \geq \frac{\eta_0}{Ck^2\mu^k} - 1, \tag{5.10}$$

and

$$\frac{\alpha_k + 1}{k} \geq \frac{\eta_0}{Ck^3\mu^k},$$

hence $(\alpha_k + 1)/k \to \infty$ as $k \to \infty$. Consequently, the rate of messages sent in the network vanishes, namely

$$\lim_{t\to\infty} \frac{\sum_{k=1}^{t} N(k)}{t} = 0.$$

□

Three main factors in the above algorithm cause the algorithm to be centralized: the global scalar $e(t)$, the global scalar $\eta(t)$, and the broadcast signal. In the following sections, we will present a decentralized algorithm inspired by the centralized one, but all global scalars are changed to local ones, and the nodes are not able to send a broadcast signal to trigger an iteration.

5.4.2 Decentralized Environment

5.4.2.1 Modified Settings

The analysis of the system becomes more complicated when we deal with the decentralized scenario. Each node works independently. We keep the assumption of synchronous operation, but the decision to transmit or not is local, so a node can be silent, while its neighbor is not. In this scenario, even the convergence of the system might not be guaranteed and we see that within an iteration, some nodes will be transmitting and others will be silent. This can cause instability in the network because the average of the estimates at every iteration is now not conserved (this is an important property of the standard consensus protocols that can be easily checked), and the scalars $\eta(k)$ and $e(k)$ defined in the previous subsection are now vectors $\eta(k)$ and $\mathbf{e}(k)$ where $\eta_i(k)$ and $e_i(k)$ are the values corresponding to a node i and are local to every node. To conserve the average in the decentralized setting, $\mathbf{e}(k)$ must take part in the state equation as we will show in what follows.

5.4.2.2 System Equation

In our approach, we consider a more general framework for average consensus where we study the convergence of the following equation:

$$\mathbf{x}(k+1) + \mathbf{e}(k+1) = W\mathbf{x}(k) + \mathbf{e}(k). \tag{5.11}$$

Some work has studied the following equation as a perturbed average consensus and considered $\mathbf{e}(k)$ to be zero mean noise with vanishing variance (see [XBK07, HDM05]). However, in our model, we consider $\mathbf{e}$ as a deterministic part of the state of the system and not a random variable. We consider sufficient conditions for the system to converge, and we use these conditions to design an algorithm that can reduce the number of messages sent in the network.

In the standard consensus algorithms, the state of the system is defined by the state vector $\mathbf{x}$, but in the modified system, the state equation is defined by the couple $\{\mathbf{x}, \mathbf{e}\}$. In the following we present a key theorem for the convergence in a decentralized setting.

Theorem 3. *Consider a system governed by the equation (5.11), let $F(k) = F(k, \mathbf{x}(k), \mathbf{x}(k-1))$ be a matrix that depends on the iteration k and two history state vectors $\mathbf{x}(k)$ and $\mathbf{x}(k-1)$. Suppose that $\mathbf{e}(k+1) = \mathbf{e}(k) - F(k)\mathbf{x}(k)$ and assume the following conditions on the matrices $A(k) = W + F(k)$ and $F(k)$:*

(a) $a_{ij}(k) \geq 0$ for all i, j, and k, and $\sum_{j=1}^{n} a_{ij}(k) = 1$ for all i and k,

(b) Lower bound on positive coefficients: there exists some $\alpha > 0$ such that if $a_{ij}(k) > 0$, then $a_{ij}(k) \geq \alpha$, for all i, j, and k,

(c) Positive diagonal coefficients: $a_{ii}(k) \geq \alpha$, for all i, k,

(d) Cut-balance: for any i with $a_{ij}(k) > 0$, we have j with $a_{ji}(k) > 0$,

(e) $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star \Rightarrow \lim_{k\to\infty} F(k, \mathbf{x}(k), \mathbf{x}(k-1))\mathbf{x}(k) = \mathbf{0}$.

Then $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$ where $x'_{ave} \in [\min_j x_j(0), \max_j x_j(0)]$; if furthermore $\mathbf{e}(0) = \mathbf{0}$ and $e_i(k) < \eta$ for all i and k, then $|x_{ave} - x'_{ave}| < \eta$.

Proof. Let us first prove that $\mathbf{x}(k)$ converges. By substituting the equation of $\mathbf{e}(k+1)$ in (5.11), we obtain:

$$\mathbf{x}(k+1) = A(k)\mathbf{x}(k), \tag{5.12}$$

where $A(k) = W + F(k)$. From the conditions (a),(b),(c), and (d) on $A(k)$, we have from [HT11] that $\mathbf{x}$ converges, i.e., $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star$. Since the system is converging, then from equation (5.11), we can see that:

$$\begin{aligned}
\mathbf{x}^\star &= W\mathbf{x}^\star + \lim_{k\to\infty} (\mathbf{e}(k) - \mathbf{e}(k+1)) \\
&= W\mathbf{x}^\star + \lim_{k\to\infty} F(k)\mathbf{x}(k) \\
&= W\mathbf{x}^\star.
\end{aligned}$$

Therefore, $\mathbf{x}^\star$ is an eigenvector corresponding to the highest eigenvalue ($\lambda_1 = 1$) of W. So we can conclude that $\mathbf{x}^\star = x'_{ave}\mathbf{1}$ where x'_{ave} is a scalar (Perron-Frobenius theorem).

The condition $\mathbf{1}^T W = \mathbf{1}^T$ on the matrix W in equation (5.2) leads to the preservation of the average in the network, $\mathbf{1}^T\mathbf{x}(k) = nx_{ave}\ \forall k$. This condition is not necessary satisfied by $A(k)$, so let us prove now that the system preserves the average x_{ave}:

$$\mathbf{1}^\top(\mathbf{x}(k+1) + \mathbf{e}(k+1)) = \mathbf{1}^\top(W\mathbf{x}(k) + \mathbf{e}(k)) \tag{5.13}$$

$$= \mathbf{1}^\top(\mathbf{x}(k) + \mathbf{e}(k)). \tag{5.14}$$

The last equality comes from the fact that W is sum preserving since $\mathbf{1}^\top W = \mathbf{1}^\top$.

Finally by a simple recursion we have that $\mathbf{1}^\top(\mathbf{x}(k) + \mathbf{e}(k)) = \mathbf{1}^\top\mathbf{x}(0) = nx_{ave}$, and the average is conserved. Moreover, since $|e_i(k)| \leq \eta$ for all i and k we have:

$$|(1/n)\mathbf{1}^\top\mathbf{x}(k) - x_{ave}| \leq \eta\ \forall k. \tag{5.15}$$

But we just proved that $\lim_{k\to\infty}\mathbf{x}(k) = x'_{ave}\mathbf{1}$, so this consensus is within η from the desired x_{ave}:

$$|x'_{ave} - x_{ave}| \leq \eta. \tag{5.16}$$

This ends the proof. □

In the decentralized environment, we gave the conditions for the system to converge. In the following section we will design an algorithm that satisfies these conditions and needs only local communications.

5.4.3 Message Reducing Algorithm

We try to solve the termination problem through a *fully decentralized* approach. We consider large-scale networks where nodes have limited resources (in terms of power, processing, and memory), do not use any global estimate (e.g. diameter of the network or number of nodes), keep only one iteration history, and can only communicate with their neighbors. Our goal is to reduce the number of messages sent while guaranteeing that the protocol converges within a given margin from the actual average.

The main idea is that a node, say i for example, will compare its new calculated value with the old one. According to the change in the estimate, i will decide either to broadcast its new value or not to do so. We divide an iteration into two parts, in the first part of the iteration, only nodes with significant change in their estimates are allowed to send messages. However, in the second part of the iteration, only nodes polled by their neighbors from phase 1 are allowed to send an update.

Before starting the linear iterative equation, nodes will select weights as in the standard consensus algorithm. The weight matrix considered here must be doubly stochastic with $0 < \alpha < w_{ii} < 1 - \alpha < 1$ for some constant α. Each node i in the network keeps two state values at iteration k:

Algorithm 1: Termination Algorithm -node i- Phase 1

$\{x_i(k), e_i(k)\}$ are the state values of node i at iteration k, $0 < \alpha < w_{ii} < 1 - \alpha < 1$, $counter_i = 1$ is the counter for the number of transmissions so far. $\eta_i(1) = \eta/2 \in \mathbb{R}$, T_k is set of $Transmit$ state. W_k set corresponding to $Wait$ state. Initially we have $T_k = W_k = \varnothing$. Every node i follows the following algorithm at iteration k.
$y_i(k+1) \leftarrow w_{ii}x_i(k) + \sum_{j \in N_i} w_{ij}x_j(k)$
$d_i \leftarrow y_i(k+1) - x_i(k) + e_i(k)$
if $|d_i| < \eta_i(counter_i)$ **then**
  i changes to a *Wait* state. $\setminus\setminus\ i \in W_k$
else
  $counter_i = counter_i + 1$
  $\eta_i(counter_i) = \eta_i(counter_i - 1) + \eta_i(1)/counter_i^2$
  $c_i \leftarrow \frac{\alpha}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big)$
  if $|c_i| \leq |e_i(k)|$ **then**
    $x_i(k+1) \leftarrow y_i(k+1) + \text{sign}(c_i.e_i(k))c_i$
    $e_i(k+1) \leftarrow e_i(k) - \text{sign}(c_i.e_i(k))c_i$
  else
    $x_i(k+1) \leftarrow y_i(k+1) + e_i(k)$
    $e_i(k+1) \leftarrow 0$
  end if
  i changes to a *Transmit* state. $\setminus\setminus\ i \in T_k$
  Notify the neighbors having maximum and minimum values.
end if
Go to Phase 2

- $x_i(k)$: the estimate of node i used in the iterative equations by the other nodes.

- $e_i(k)$: a real value that monitors the shift from the average due to the iterations where node i did not send a message to its neighbors. It is initially set to zero, $e_i(0) = 0$.

Each node also keeps its own boundary threshold $\eta_i(k)$ where $\eta_i(1) = \frac{\eta}{2} = constant\ \forall i$. Note that $\eta_i(k)$ is increased after every transmission as in the centralized case, but the difference here is that it is local to every node.

Each iteration is divided into two phases:
In the first phase, a node i can be in one of the two following states:

- *Transmit*: The set of nodes corresponding to this state is T_k, where the subindex k corresponds to the fact that the set can change with every iteration k. The nodes in T_k send their new calculated estimate to their neighbors. They also poll the nodes having maximum and minimum estimates in their neighborhood to transmit in phase 2.

Algorithm 2: Termination Algorithm - Phase 2

1: $\{x_i(k), e_i(k)\}$ are the state values of node i at iteration k.
2: **for all** nodes i having *Wait* state **do**
3: $y_i(k+1) \leftarrow w_{ii}x_i(k) + \sum_{j \in N_i} w_{ij}x_j(k)$
4: **if** i received a poll message from any neighbor **then**
5: $z_i(k+1) \leftarrow (w_{ii} + \sum_{j \in N_i \cap W_k} w_{ij})x_i(k) + \sum_{j \in N_i \cap T_k} w_{ij}x_j(k)$
6: $x_i(k+1) \leftarrow z_i(k+1)$
7: $e_i(k+1) \leftarrow y_i(k+1) - z_i(k+1) + e_i(k)$
8: i changes to a $Cut - Balance$ state. $\setminus\setminus\ i \in B_k$
9: **else**
10: $x_i(k+1) \leftarrow x_i(k)$
11: $e_i(k+1) \leftarrow y_i(k+1) - x_i(k) + e_i(k)$
12: i changes to a *Silent* state. $\setminus\setminus\ i \in S_k$
13: **end if**
14: **end for**
15: $k+1 \leftarrow k$

- *Wait*: The set of nodes corresponding to this state is W_k. The node's decision will be taken in the second phase of the iteration based on the action of nodes in the *Transmit* state (depending if they were polled by any of their neighbors).

In the second part of the iteration, nodes that are in W_k will be classified as follows:

- *Silent*: The set of nodes corresponding to this state is S_k. These are the nodes that will remain silent with no message sent from their part in the network. The nodes in S_k have that non of their neighbors sending them any poll message.

- *Cut-Balance*: The set of nodes corresponding to this state is B_k. They are called $Cut - Balance$ because they insure the cut-balance condition (d) of Theorem 3. They are the nodes in W_k that have been polled by at least one neighbor in T_k .

The two phases of the termination protocol implemented at each node are described by pseudocode in Algorithm 1 and 2. Nodes in the T_k set (the set of nodes that are in a $Transmit$ state) will broadcast their estimate to their neighbors at the end of the first phase, while nodes in W_k set (or $Wait$ state) will postpone their decision to send or not till the next phase. Nodes that do not receive a message from their neighbors at a certain iteration, use the last seen estimate from the specified neighbors (note: absence of messages from a neighbor during an iteration does *not* mean the failure of link, it means that the neighbor is broadcasting the same old estimate as before, so we may differentiate the link failure by a "keep alive" message sent frequently to maintain connectivity and set of neighbors). The **input** for the algorithm are the estimates of the neighbor of i, the weights selected for these neighbors, and the state values $\{x_i(k), e_i(k)\}$. The **output** of the first phase is the new

state values $\{x_i(k+1), e_i(k+1)\}$ for nodes in T_k and the output of the second phase is the new state values $\{x_i(k+1), e_i(k+1)\}$ for nodes in W_k. Let us go through the lines of the algorithm. In phase 1, $y_i(k+1)$ of line 2 is the weighted average of the estimates received by node i; without the termination protocol this value would be sent to all its neighbors. The protocol evaluates how much $y_i(k+1)$ differs from the state value $x_i(k)$. This difference accumulates in d_i in line 3. If this shift is less than a given threshold η_i, the node will wait for next phase to take decision. If the condition in line 4 is not satisfied, that means the node will send a new value to its neighbors. Lines $7-8$ concerns the extending of the boundary threshold $\eta_i(k)$ after every transmission. Note that by this extension method, we have $\eta_i(k) < \eta \ \forall i, k$ since

$$\lim_{k\to\infty} \eta_i(k) = \eta_i(1)(\sum_{i=1}^{\infty} 1/k^2) < \eta_i(1) \times 2 = \eta.$$

We introduce in line 9 a new scalar c_i used for deciding which portion of $e_i(k)$ the node will send in the network. In lines $11-12$ and $14-15$, the algorithm satisfies the equation (5.11). Then the new state value $x_i(k+1)$ is sent to the neighbors and $e_i(k+1)$ is updated accordingly. In Phase 2 of the algorithm (Algorithm 2), nodes initially in the wait state will decide either to send a cut-balance massage or to remain silent, the cut balance messages are sent when a node receives a poll message from any of its neighbors.

5.4.4 Convergence Study

The convergence of the previous algorithm is mainly due to the fact that the proposed algorithm satisfies the conditions of convergence given in 5.4.2.2. In fact, the algorithm is designed to satisfy all these conditions that guarantee convergence. Starting with the state equation, we can notice from the Algorithm 1, given whatever the condition the nodes face, that the sum of the new generated state values $\{x_i(k+1), e_i(k+1)\}$ is as follows:

$$x_i(k+1) + e_i(k+1) = y_i(k+1) + e_i(k),$$

where $y_i(k+1) = w_{ii}x_i(k) + \sum_{j\in N_i} w_{ij}x_j(k)$. As a result the system equation is the one studied in section 5.4.2.2 (equation (5.11)). Before going through the different conditions in the Theorem 3, we should show that according to the algorithm given in pseudo code, $\mathbf{e}(k+1) = \mathbf{e}(k) - F(k)\mathbf{x}(k)$ for some matrix $F(k)$ such that $F(k)\mathbf{1} = \mathbf{0}$. From the algorithm we can write,

$$e_i(k+1) = e_i(k) - v_i(k), \tag{5.17}$$

where $v_i(k)$ differs according to the state of the node i, but it only depends on the estimate of node i and its neighbors:

$$v_i(k) = \begin{cases} \pm\frac{\alpha}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big) & \text{if } i \in T_k - (1), \\ \pm\frac{\alpha\gamma}{(1-w_{ii})}\Big(y_i(k+1) - x_i(k)\Big) & \text{if } i \in T_k - (2), \\ x_i(k) - y_i(k+1) & \text{if } i \in S_k, \\ z_i(k+1) - y_i(k+1) & \text{if } i \in B_k, \end{cases} \tag{5.18}$$

where $T_k - (1)$ is the set of nodes subset in T_k where $|c_i| \leq |e_i(k)|$, and $T_k - (2)$ set of nodes where $|c_i| > |e_i(k)|$. In the latter case, $e_i(k+1) = 0$, but we can always find $\gamma < 1$ such that $e_i(k+1) = e_i(k) - \gamma(\text{sign}(c_i.e_i(k)c_i) = 0$ where $c_i = \frac{\alpha}{(1-w_{ii})}(y_i(k+1) - x_i(k))$. $y_i(k+1)$ and $z_i(k+1)$ are as indicated in the algorithm and are a linear combination of the elements of $\mathbf{x}(k)$. From the equation of $v_i(k)$, we can also see that it is a linear combination of the elements of $\mathbf{x}(k)$, such that the coefficients sum to 0. A row i in $F(k)$ will be the coefficients of the estimates $\mathbf{x}(k)$ in $v_i(k)$, so $F(k)\mathbf{1} = \mathbf{0}$.

Now we can study the conditions mentioned in the Theorem 3 on the matrix $A(k) = W + F(k)$.

Lemma 11. *$A(k)$ is a stochastic matrix that satisfies conditions (a),(b),and (c) of Theorem 3.*

Proof. First, we can see that $A(k)\mathbf{1} = \mathbf{1}$ since $W\mathbf{1} = \mathbf{1}$ and $F(k)\mathbf{1} = \mathbf{0}$. It remains to prove that all entries in the matrix $A(k)$ are non negative. We will prove this by considering each row i of $A(k)$ according to the action taken by node i. We can distinguish four cases:

1. Node $i \in T_k$ - condition 1: $|c_i| \leq |e_i(k)|$
   $a_{ii} = w_{ii} - \frac{\alpha}{1-w_{ii}} \times (1 - w_{ii}) = w_{ii} - \alpha > 0$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} + \frac{\alpha}{1-w_{ii}} \times w_{ij} \geq w_{ij} > 0 \quad \forall j \in N_i$.
   or
   $a_{ii} = w_{ii} + \frac{\alpha}{1-w_{ii}} \times (1 - w_{ii}) > \alpha$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} - \frac{\alpha}{1-w_{ii}} \times w_{ij} > 0 \quad \forall j \in N_i$ since $\alpha < w_{ii} < 1 - \alpha$.

2. Node $i \in T_k$ - condition 2: $|c_i| > |e_i(k)|$
   since $|c_i| > |e_i(k)|$, we can always find positive $\gamma < 1$, such that
   $a_{ii} = w_{ii} - \frac{\gamma\alpha}{1-w_{ii}} \times (1 - w_{ii}) = w_{ii} - \gamma\alpha > 0$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} + \frac{\gamma\alpha}{1-w_{ii}} \times w_{ij} \geq w_{ij} > 0 \quad \forall j \in N_i$.
   or
   $a_{ii} = w_{ii} + \frac{\gamma\alpha}{(1-w_{ii})} \times (1 - w_{ii}) > \alpha$ since $w_{ii} > \alpha$
   and $a_{ij} = w_{ij} - \frac{\gamma\alpha}{1-w_{ii}} \times w_{ij} > (1 - \frac{\alpha}{\alpha})w_{ij} > 0 \quad \forall j \in N_i$.

3. Node $i \in S_k$:
   then $a_{ii} = w_{ii} + (1 - w_{ii}) = 1$
   and $a_{ij} = 0 \quad \forall j \neq i$.

4. Node $i \in B_k$:
   then $a_{ii} \geq w_{ii} > 0$
   and $a_{ij} \in \{w_{ij}, 0\} \quad \forall j \neq i$.

Therefore, $A(k)$ is stochastic at every iteration k. □

Definition 5. *Two matrices, A and B, are said to be equivalent with respect to a vector* $\mathbf{v}$ *if and only if* $A\mathbf{v} = B\mathbf{v}$.

Notice that $A(k)$ satisfies conditions (a),(b), and (c) of Theorem 3, but possibly not the cut balance condition (d) because for a node $i \in T_k$ that transmits, $a_{ij}(k) > 0 \ \forall \ j \in N_i$, but it can be that $\exists j \in N_i$ such that $a_{ji} = 0$ if j was silent at that iteration ($j \in S_k$). However, the next lemma shows that there is a matrix $B(k)$ equivalent to $A(k)$ with respect to $\mathbf{x}(k)$ that satisfies all the conditions.

Lemma 12. *For all k, there exists a matrix* $B(k)$ *equivalent to* $A(k)$ *with respect to* $\mathbf{x}(k)$ *such that* $B(k)$ *satisfies the conditions (a),(b),(c), and (d) of Theorem 3.*

Proof. Let $m(k) = \text{argmin}_{j \in N_i} x_j(k)$ and $M(k) = \text{argmax}_{j \in N_i} x_j(k)$, we will proof the existence of $B(k)$ by modifying $A(k)$ in such a way to preserve the properties (a) to (c) and to add the new property (d). For simplicity of notation we will drop k from the variables since this is true for every k. Note first that the condition (d) is not satisfied in A only for the rows and columns where i belongs to T_k (because in this case $a_{ij} > 0$ for $j \in N_i$, but a_{ji} could be 0 if j is in a silent state S_k). Let $\mathbf{a}_i$ denote the row i of A and $\mathbf{b}_i$ denote the row i of B. Since any node i in T_k must poll the nodes m and M to transmit in Phase 2, then we are sure that the column i has at least three non zero elements (a_{ii}, a_{mi}, and a_{Mi}). Let $C_i = \{j \mid a_{ji} > 0 \ , \ i \neq j\}$, so we are sure that C_i contains at least two elements m and M. The cut balance condition requires that the row of i must only have positive values at the index where the column is positive. We can write that

$$\begin{aligned}
\mathbf{a}_i\mathbf{x}(k) &= a_{ii}x_i(k) + \sum_{j \in N_i} a_{ij}x_j(k) \\
&= a_{ii}x_i(k) + \sum_{j \in C_i} a_{ij}x_j(k) + \sum_{j \in N_i - C_i} a_{ij}x_j(k) \\
&= a_{ii}x_i(k) + \sum_{j \in C_i} a_{ij}x_j(k) + hx_M(k) + fx_m(k), \qquad (5.19)
\end{aligned}$$

where $h = \frac{\sum_{j \in N_i - C_i} a_{ij}(x_j - x_m)}{x_M - x_m}$, $f = \frac{\sum_{j \in N_i - C_i} a_{ij}(x_M - x_j)}{x_M - x_m}$. Since $h \geq 0$, $f \geq 0$ and $f + h = \sum_{j \in N_i - C_i} a_{ij}$, we can define the row vector $\mathbf{b}_i$ to be:

$$\mathbf{b}_i := \begin{cases} b_{ij} = a_{ij} & \text{if } j = i, \\ b_{ij} = a_{ij} + f & \text{if } j = m, \\ b_{ij} = a_{ij} + h & \text{if } j = M, \\ b_{ij} = a_{ij} & \text{if } j \in C_i - m - M, \\ b_{ij} = 0 & \text{if } j \in N_i - C_i. \end{cases} \qquad (5.20)$$

Finally, $\mathbf{b}_i\mathbf{x}(k) = \mathbf{a}_i\mathbf{x}(k)$ and it satisfies the conditions (a) to (c), so $\forall i \in T_k$, we replace $\mathbf{a}_i$ by $\mathbf{b}_i$ and we get the new matrix B which is equivalent to A with respect to $\mathbf{x}(k)$. □

Lemma 13. *The message reduction algorithm (Phases 1, 2) satisfies condition (e) of Theorem 3.*

Proof. We will prove it by contradiction. Suppose $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^\star$, but $\lim_{k\to\infty} F(k)\mathbf{x}(k) \neq \mathbf{0}$, then there exists a node i such that $\lim_{k\to\infty} x_i(k) = x_i^\star$ and $\lim_{k\to\infty} y_i(k+1) = w_{ii}x_i^\star + \sum_{j\in N_i} w_{ij}x_j^\star = y_i^\star$, but $y_i^\star - x_i^\star = \delta^\star > 0$. From Algorithm 1, we can see that the node will enter a transmit state infinitely often (because d_i increases linearly with δ^* and it will reach the threshold η_i). Then, the node i will update its estimate according to the equation

$$x_i(k+1) = y_i(k+1) + \frac{\alpha}{1-w_{ii}}(y_i(k+1) - x_i(k)).$$

Letting $k \to \infty$ yields

$$(1 + \frac{\alpha}{1-w_{ii}})\delta^\star = 0.$$

Thus, $\delta^\star = 0$ which is a contradiction, and the algorithm satisfies condition (e) of Theorem 3. □

The algorithm also provides that $|e_i(k)| \leq \eta_i(k)\ \forall k, i$ and $\eta_i(k) < \eta\ \forall i, k$, as in the first phase this condition is satisfied by construction, and for the second phase of the iteration, nodes from Phase 1 can check for worst case analysis and they only enter into *Wait* state if they are sure that the condition can be satisfied in the next phase iteration.

Now we are ready to state the main Theorem in this section:

Theorem 4. *The nodes applying the message reducing algorithm given in pseudocode by Algorithm 1 and 2, have estimates converging to a consensus within a margin η from x_{ave}, i.e., $\lim_{k\to\infty} \mathbf{x}(k) = x'_{ave}\mathbf{1}$ and $|x'_{ave} - x_{ave}| \leq \eta$.*

Proof. The theorem is due to the fact that the Lemmas given in this subsection show that the algorithm satisfies all the convergence conditions of Theorem 3. □

As a result, the convergence of nodes' estimates of the distributed algorithm for message reduction is guaranteed. We study in the next section the performance of this algorithm on random networks, we also address the case of faulty unreliable links, and we show the stability of the algorithm in the presence of nodes changing their estimate possibly due to faulty estimates or due to a changing environment.

5.4.5 Simulations

The termination algorithm (the message reduction algorithm) is simulated on two types of random graphs, the Random Geometric Graphs (RGG) and the Erdos Renyi

(ER) graphs. To measure the distance from the average, we consider the *normalized error* metric defined as,

$$\bar{L}_k = \frac{||\mathbf{x}(k) - \bar{\mathbf{x}}||_2}{||\mathbf{x}(0) - \bar{\mathbf{x}}||_2},$$

where the vector $\bar{\mathbf{x}}$ is $\bar{\mathbf{x}} = x_{ave}\mathbf{1}$. For example when $\log(\bar{L}_k) = -3$, that means the error became 0.1% of the initial one. Initially, each node has a uniformly random value between 0 and 10. On the RGG with 500 nodes and connectivity radius 0.093, Fig. 5.3 gives a comparison between standard average consensus algorithms and the termination algorithm proposed in this chapter. The standard algorithm referred here is the one that uses Eq. (5.1) without applying any attempts to terminate the protocol. The figure shows the effect of varying the precision η in the termination algorithm on the number of messages (active nodes per iteration). In the study of the convergence of the algorithm, we showed that the algorithm converges to at most η from the true average. As the figure shows, with termination algorithm the error converges to a value x'_{ave} different from the real average x_{ave}, smaller η gives closer estimate to x_{ave} but more messages are sent. In fact, the termination algorithm passes through three phases: the *first* phase is the initial start where nodes usually have large differences in their estimates and they tend to send many messages while decreasing the error (same start as standard algorithm), the *second* phase is the most efficient phase where nodes saves messages while continuing to decrease the error. The *final* phase is the stabilizing phase where nodes converge to a value close to the true average. The start and duration of each phase depends on the value of η. Similar results were given on ER graphs.

Links in networks (specially wireless networks) can be unreliable. The algorithm being totally decentralized, and uses only one history estimate can be applied for the dynamic scenario. The weight matrix is then dynamic and at every iteration k a different $W(k)$ is considered and constructed locally as following. Before starting the algorithm we let $W(0)$ be generated locally satisfying convergence conditions as throughout the chapter. At iteration k, a weight on a link can take two values, the original weight ($w_{ij}(k) = w_{ij}(0)$) if link $l \sim \{i, j\}$ is active or $w_{ij}(k) = 0$ if the link failed. When there are failures of links, some weight is added to the self-weight of nodes to preserve the double stochastic property of the matrix $W(k)$. In Fig. 5.4, we consider the RGG of 100 nodes and connectivity radius 0.19 with unreliable links. $\eta = 0.01$ is fixed for both graphs (the graph with the high link failure probability graph and the low link failure probability one). With high link failure probability, the network sends less messages because there are less links in the network, but the speed to consensus is slower than that of the low failure probability. Note that non of the synchronous termination algorithms given in the related work consider a dynamic topology.

In some scenarios, we are interested in a rapid detection of a sudden change in the true average due to the environment change. In the real life, if sensor nodes are measuring the temperature of a building, and the temperature changed largely (probably due to a fire in a certain room in the building), fast detection of this change can be very useful. In Fig. 5.5, we assumed that at certain iterations some

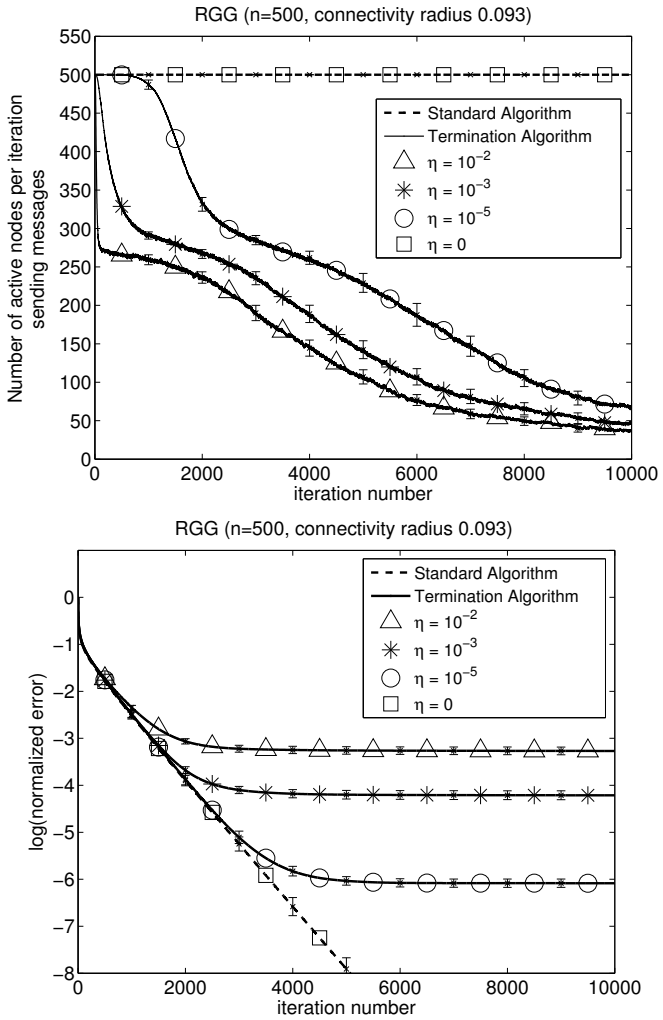

Figure 5.3: (a) The number of active nodes at every iteration. (b) The error from the average at every iteration. This shows the effect of η in the message reduction algorithm on RGG networks. In standard algorithms, nodes' estimates converge to the real average, so the error decreases linearly, but nodes are not aware of how close they are to consensus, so they are all always active sending messages. With termination algorithm, nodes converge to a value at most η away from the real average, different values of η give different precision error. The algorithm gives a trade-off between precision and number of messages (The standard algorithm is just a special case of termination algorithm for $\eta = 0$).

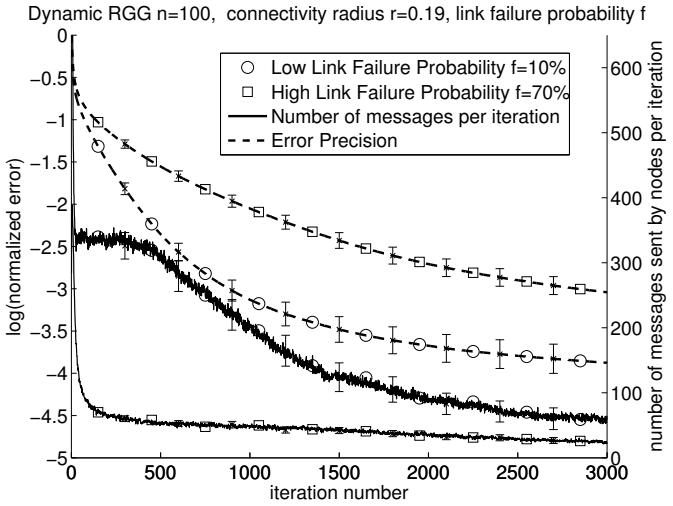

Figure 5.4: Termination algorithm on a dynamic RGG with different link failure probabilities. On low link failure probability graphs, the messages are less than that of the high failure probability, but the convergence speed is slower.

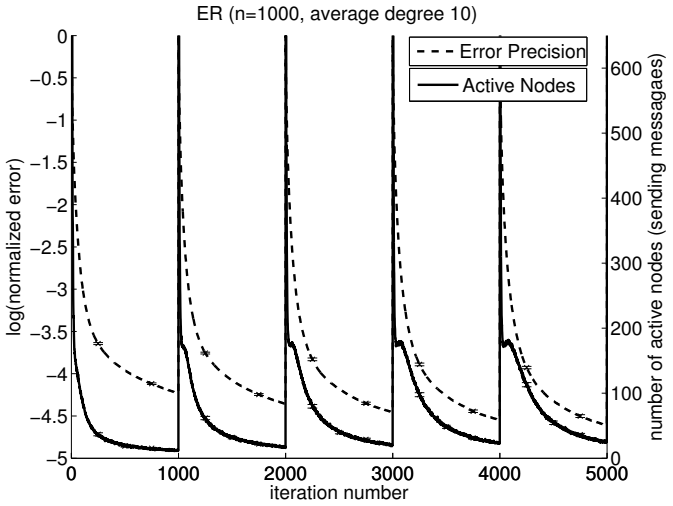

Figure 5.5: Normalized error and number of nodes transmitting on the ER graphs with 1000 nodes, where every 1000 iterations random 10% of the nodes change their estimates. The figure shows the self-adaptive feature of the algorithm: when the algorithm has already converged to a stable state and is communicating little, and an exogenous event pushes the value in a node away from the current average, the algorithm detects the change and ramps up its communication intensity.

nodes for a certain reason change their estimates to a completely different one. We considered an Erdos Renyi graph with 1000 nodes and average degree 10, as an initial state, nodes estimate takes a value in the interval $[0, 10]$ uniformly at random. Every 1000 iterations, on average, 10% of the nodes restart there estimate by a new one in the interval $[10, 20]$ chosen uniformly at random. The figure shows the self-adaptive feature of the algorithm: when the algorithm has already converged to a stable state and is communicating little, and an exogenous event pushes the value in a node away from the current average, the algorithm detects the change and ramps up its communication intensity and stabilizes again. So the algorithm is able to cope with the sudden change and the system automatically adapts its behavior. With every change in the estimates, the network give a burst of messages to stabilize the network to the new average.

5.5 Conclusion

In this chapter, we give an algorithm to reduce the messages sent in average consensus. The algorithm is totally decentralized and does not depend on any global variable, it only uses the weights selected to neighbors and one iteration history of the estimates to decide to send a message or not. We proved that this algorithm is converging to a consensus at most η from the true average. The algorithm can be applied on dynamic graphs and is also robust and adaptive to errors caused by a node suddenly changing its estimate.

Chapter 6

Graph Clustering by Random Walks

Contents

A community of nodes (or a cluster of nodes) in a network is a group of vertices that are well connected to each other, but are less connected with the remaining part of the network. Detecting clusters in networks has many applications. Communities in social networks are formed by people having common interest. Clusters in the web graph can group pages with similar topics. E-commerce, classification, computer vision, bioinformatics, and machine learning are only few areas of application of network clustering.

Comparing different possible graph clustering outputs and selecting the best outcome are carried out by introducing a quality metric that serves as an objective function. There is still no consensus in the literature on which quality metric for graph clustering is the best one. In this chapter, we introduce a new fitness measure for evaluating a clustering algorithm based on random walks' properties. Roughly speaking, our fitness index is higher the faster a random walk constrained to the cluster reaches its stationary distribution and the slower it escapes from the cluster in the unconstrained case. Both effects can be quantified considering the eigenvalues of appropriate matrices. Beside introducing this new metric, we propose a randomized algorithm for clustering the network accordingly. The algorithm is *local* because it relies only on a partial view of the entire network. In particular, if the graph represents the topology of a network where nodes have computing capabilities, the algorithm can run in parallel at each node without the need of a central unit. Being local, clusters can be formed in parallel and the computation complexity is distributed among clusters. The algorithm can also find small clusters that are more difficult to be detected by the global clustering methods.

6.1 Related Work

One of the most used metrics is the modularity [NG04] which gives a score to the cluster by comparing the number of edges falling inside the clusters with the number of edges of a random graph having similar characteristic as the original one. Although the modularity is widely used in applications, it is shown that it cannot distinguish small clusters having links of order $O(\sqrt{m})$ where m is the total number of links [FB07]. The silhouette index [TSK05] uses distances between the nodes presented in the cluster and those outside it, its drawback being its high computational cost as it requires to compute the shortest path between all node pairs. Another approach [KVV00] evaluates a clustering score by using the concept of inter-cluster conductance, but it ignores internal cluster density. Some graph partitioning algorithms based on PageRank vectors of a graph have been proposed in [ACL06] to find a cut with a certain conductance in the graph. All these metrics turn out to be biased toward large communities [AGMZ11]. Many practical algorithms have been proposed as hierarchical clustering [KM86], Markov clustering [VD08], bisecting K-means, and spectral clustering [KVV00]. Their drawback is that they are global clustering methods which require as input the entire graph to calculate the clustering. Moreover their output is biased toward equal size clusters (so small communities tend to disappear using these algorithms). A complete survey of fitness measures and clustering is given in [Sch07].

6.2 Notation

Consistent with the notation along this thesis, $G = (V, E)$ is an undirected unweighted connected graph without self-loops, where $V = \{1, \ldots, n\}$ is the set of vertices and $E = \{1, \ldots, m\}$ is the set edges. Let $d_G(i) = |\{j \in V, (i,j) \in E\}|$ be the degree of a node i in G, D_G be a diagonal matrix having on its diagonal the degree of the nodes in G and let A_G be the adjacency matrix of the graph G where $a_{ij} = 1$ if $(i,j) \in E$, and $a_{ij} = 0$ otherwise. For any set $S \subseteq V$, let $D_G(S)$ (resp. $A_G(S)$) be the sub-matrix of D_G (resp. A_G) obtained considering only rows and columns corresponding to the vertices in S. Let $G(S) = (S, E(S))$ be the subgraph induced by $S \subseteq V$ where $E(S) = \{(i,j) \in E | i,j \in S\}$. Observe that in general $D_G(S) \neq D_{G(S)}$ because $D_G(S)$ contains the degree of nodes in the original graph G which are different from their degrees in the induced subgraph $G(S)$. Conversely, $A_G(S) = A_{G(S)}$ as the adjacency matrix is not changed. If P is a substochastic matrix (a square matrix with nonnegative entries so that every row adds up to at most 1), let $\rho(P)$ be the largest eigenvalue in module of P. When P is stochastic, let $s(P) = 1 - \mu(P) \in [0,1]$ be its spectral gap[1] where $\mu(P)$ is the second largest eigenvalue in module of P.

A clustering $\mathcal{C}_G$ of a graph G is a partition of the vertices such that $\mathcal{C}_G = \{C_1, \ldots C_k\}$ where $C_1 \cup \ldots \cup C_k = V$ and $C_u \cap C_v = \phi$ for all clusters C_u and C_v in $\mathcal{C}_G$.

[1] If P is a scalar, we consider $s(P) = 1$ by convention.

Let $C(i) = \{C_u \in \mathcal{C}_G; i \in C_u\}$ be the cluster that contains node i. Let $f : V \to \mathbb{R}$ be a scoring function for the nodes, define a cluster score $f(C_u) = \sum_{i \in C_u} f(i)$, and a clustering algorithm score $f(\mathcal{C}_G) = \sum_{u=1}^{k} f(C_u) = \sum_{i=1}^{n} f(i)$.

6.3 The Random Walk Fitness Measure

In this section, we introduce a new scoring function $f(.)$ that can serve as a quality measure for a clustering algorithm. A good clustering algorithm identifies clusters that are well connected internally, but weakly connected with the rest of the network. Inspired by this intuitive definition, the function f should have the following properties:

1. A cluster whose induced subgraph is disconnected should receive the minimum score.
2. A clique graph clustered as a single cluster should have the highest score among all clusterings for graphs with the same number of nodes.
3. For a given clustering, adding links within clusters should increase the score while removing them should only decrease the score.
4. For a given clustering, adding links between different clusters should decrease the score while removing them should increase the score.
5. Within a cluster, the higher the degree of a node, the more it contributes to the score.
6. Boundary nodes in a cluster that have links to other clusters have less score than internal nodes.

Given a graph clustering $\mathcal{C}_G = \{C_1, ..., C_k\}$, we define the score of a vertex $i \in V$ is given by

$$f(i) = \alpha_i \times s_{C(i)} \times \rho_{C(i)},$$

where $s_{C(i)}$ quantifies how fast a random walk on $G(C(i))$ (and then constrained to the cluster $C(i)$) reaches its steady state distribution, $\rho_{C(i)}$ corresponds to the probability that a random walk on the whole graph G that starts inside the cluster $C(i)$ keeps staying inside the cluster at a following step (see below for a more formal definition), and finally α_i differentiates among different nodes in the same cluster according to the last two properties. Given this definition of the scoring function, the score of cluster C_u is:

$$f(C_u) = \left(\sum_{i \in C_u} \alpha_i \right) s_{C_u} \rho_{C_u}.$$

Below we define formally the different quantities α_i, s_{C_u} and ρ_{C_u} and show that $f(.)$ satisfies the required properties of a good clustering function.

First, we define s_{C_u} as

$$s_{C_u} \triangleq s\left((D_{G(C_u)} + I)^{-1}(A_{G(C_u)} + I)\right),$$

that is the spectral gap of the transition probability matrix of a simple random walk on the subgraph induced by the cluster nodes C_u adding self-loops [LO81]. This value ranges between 0 for a disconnected graph and 1 for a fully connected network (a clique). Given a random walk starting at time 0 from a node in the cluster, the difference between the probability distribution of the position of the random walker at time t and its stationary distribution can be bounded by $B(1 - s_{C_u})^t$, with B being an appropriate constant. Then the larger s_{C_u}, the faster the distribution converges to its stationary distribution, i.e., the faster the random walk *mixes*. The spectral gap of the transition probability matrix is then also a measure of how well connected the network within a cluster is. The presence of s_{C_u} as a multiplicative factor in the scoring function guarantees that the first two properties are satisfied. Moreover, due to the interlacing property of eigenvalues [CDH+05], adding more links between the nodes of the same cluster usually increases the spectral gap while removing links decreases it, which supports the third property of a good clustering function.

Second, we define

$$\rho_{C_u} \triangleq \rho\left(D_G(C_u)^{-1}A_G(C_u)\right).$$

Given that $D_G(C_u)$ considers the degrees of the nodes[2] in the original graph G, $P = D_G(C_u)^{-1}A_G(C_u)$ is a substochastic matrix. If we consider the transition probability matrix of a random walk on the whole graph G, P is the submatrix obtained by extracting only the rows and the columns corresponding to the nodes in C_u. Given a random walk on G starting at a node i in C_u, and assuming that P is a primitive matrix, it is possible to show [DS65] that the conditional probability distribution given that the random walk does not exit from C_u converges to $\pi \in [0,1]^{|C_u|}$ (we consider only the probabilities for the nodes in C_u, for all the other nodes the probability is clearly 0 under the conditioning event), that satisfies the following equation $\pi^T P = \pi^T \rho(P)$. Then $\rho(P)$ can be interpreted as the probability that at each step the random walker does not exit from C_u, given that it has already spent a long time in C_u.[3] The term ρ_{C_u} quantifies then the effect of outer links connecting the cluster C_u to other clusters. Obviously, it ranges between 0 and 1. It is equal to 1 when there is no link between nodes in C_u and nodes in $V \backslash C_u$ and then in particular when $C_u = V$ since the graph is connected. It is equal to 0 if the subgraph $G(C_u)$ has no links. Adding links between clusters can only decrease ρ while removing them can only increase it. The factor ρ_{C_u} guarantees that the fourth property is satisfied.

[2] The inverse $D_G(C_u)^{-1}$ always exists because $D_G(C_u)$ is a diagonal matrix having strictly positive diagonal values ($d_G(i) \geq 1$ because G is connected).

[3] Otherwise if we consider that the random walk initial position in C_u follows the probability distribution π, $\rho(P)$ is simply the probability that the random walker does not exit from C_u at each step.

Finally, α_i represents the contribution of a node to the final score depending on its connectivity to other clusters. To satisfy the last two properties required for the function f, the value α_i is chosen as follows:

$$\alpha_i \triangleq \frac{d_i^{in}}{1 + d_i^{out}},$$

where $d_i^{in} = d_{G(C(i))}(i)$ is the number of nodes in $C(i)$ connected to i and $d_i^{out} = d_G(i) - d_i^{in}$ is the number of nodes in $V \backslash C(i)$ connected to i.

6.4 Clustering Algorithm

The function f presented in the previous section gives a scoring mechanism to evaluate a clustering algorithm. In particular, the optimal clustering algorithm can be written as follows:

$$\underset{\mathcal{C}_G = \{C_1, ..., C_k\}}{\operatorname{argmax}} \quad f(\mathcal{C}_G). \tag{6.1}$$

Let $\mathcal{C}_G^*$ be the solution of (6.1) and $f^* = f(\mathcal{C}_G^*)$ be its value. The optimal clustering and its value are computationally difficult to find, so we will give first some bounds on the optimal value f^* and we will propose a local search clustering algorithm that can be implemented with an acceptable complexity and in a distributed way.

6.4.1 Bounds on f^*

Proposition 15. *For the clustering optimization problem (6.1), the following bounds hold for the optimal value f^*:*

$$2 \times m \times s_V \leq f^* \leq 2 \times m, \tag{6.2}$$

where s_V is the spectral gap of the simple random walk on all the graph G ($s_V = 1 - \mu\left((D + I)^{-1}(A + I)\right)$).

Proof. For any clustering $\mathcal{C}_G = \{C_1, ..., C_k\}$ of the graph G we have,

$$\begin{aligned} f(\mathcal{C}_G) &= \sum_i f(i) = \sum_i \frac{d_i^{in}}{1 + d_i^{out}} s_{C(i)} \rho_{C(i)} \\ &\leq \sum_i \frac{d_i^{in}}{1 + d_i^{out}} \leq \sum_i d_i^{in} \leq \sum_i d_G(i) \\ &= 2 \times m, \end{aligned}$$

where m is the number of links in the graph G and the first inequality follows from both s_{C_u} and ρ_{C_u} being at most equal to one. From this upper bound, it follows that $f(\mathcal{C}_G^*) \leq 2 \times m$.

The optimal clustering has a value greater than any possible clustering. Taking the graph as one cluster $\mathcal{C}_G = \{V\}$ is a valid clustering of G. Thus, a lower bound on the optimal value is given as follows:

$$f^* \geq f(\mathcal{C}_G = \{V\}) = \sum_i d_i^{in} \times s_V \times 1 = 2 \times m \times s_V,$$

where s_V is the spectral gap of the simple random walk on all the graph G ($s_V = 1 - \mu\left((D+I)^{-1}(A+I)\right)$). □

We observe that both of the bounds are tight for the fully connected graph (let us denote it K_n). Indeed nodes in K_n are grouped in a single cluster ($\mathcal{C}_G = V$) and $f(V) = 2m$ since $d_i^{out} = 0$ for any vertex i and $s_V = \rho_V = 1$.

Due to the following proposition, the subgraph induced by a cluster of the optimal clustering is connected as long as it has at least an internal link.

Proposition 16. *Let $\mathcal{C}_G^* = \{C_1, ..., C_k\}$ be an optimal clustering for a graph G, then for any $C_u \in \mathcal{C}_G^*$, if the subgraph $G(C_u)$ has at least one link, it is connected.*

Proof. We sketch a proof of the proposition by contradiction. Suppose there exists a graph whose optimal clustering $\mathcal{C}_G^*$ outputs a cluster C_u such that $G(C_u)$ has at least one link, but it is disconnected. It follows that $f(C_u) = 0$ since $s_{C_u} = 0$ for disconnected graphs. However, there is a subset of vertices $H \subset C_u$ such that $|H| \geq 2$ and $G(H)$ is connected (because there is at least one link in $G(C_u)$) and it holds $f(H) > 0$. Now if we replace C_u with two clusters H and $C_u - H$, the new clustering has a strictly higher value than $\mathcal{C}_G^*$ (contradiction). □

6.4.2 Local Search Clustering Algorithm

The optimal clustering can be computationally costly because calculating the spectral gap of a random walks has complexity $O(n^3)$. In this section, we present a local clustering algorithm that allows the clustering to be done in a distributed way. The algorithm applies the generic local search approach. Let X be the set of all possible clusterings of graph G. We define two cluster x and y belonging to X to be neighbors if and only if they differ only for a single vertex that belongs to two different clusters in x and in y. A local search algorithm for clustering operates as follows:

1. Let x be some initial clustering;
2. While there is a neighboring G-clustering y with higher score value ($f(y) > f(x)$), set $x := y$;
3. Return the final (locally optimal) solution x.

The algorithm is an iterative one. In our local clustering algorithm we follow the above steps but we add some randomness in choosing the neighbor in step two. In fact, at every iteration, a cluster, say it C_u, is chosen uniformly at random. This

Algorithm 3: Local Clustering Algorithm

$G = (V, E)$ where $V = 1...n$ and $E = 1...m$.
Initial clustering $\mathcal{C}_G^0 = \{C_1, ..., C_n\}$ where $C_i = \{i\}$.
$E_{C_u}^+ = \{(i, j) \in E | i \in C_u, j \notin C_u\}$ is the set of C_u's outgoing links.
for $k = 1 : T_{stop}$ **do**
  $\mathcal{C}_G^k = \mathcal{C}_G^{k-1}$;
  let C_u be a cluster chosen uniformly at random from $\mathcal{C}_G^k$;
  let (i, j) be a link chosen uniformly at random from $E_{C_u}^+$;
  let C_v be the cluster containing j (i.e., $C_v = C(j)$);
  C_u proposes to j to join (if it didn't yet propose to j after the last change within C_u occurred) ;
  if $f(C_u) + f(C_v) < f(C_u \cup \{j\}) + f(C_v \backslash \{j\})$ **then**
    j accepts the proposal;
    $C_u \leftarrow C_u \cup \{j\}$;
    $C_v \leftarrow C_v \backslash \{j\}$;
    if $C(j) = \phi$ **then**
      Remove C_v from $\mathcal{C}_G^k$;
    end if
  else
    j rejects the proposal;
  end if
  $k \leftarrow k + 1$;
  If all clusters don't have any more proposals **break**;
end for
return $\mathcal{C}_G^{k-1}$

random cluster selects one of the outgoing links uniformly at random and proposes to the endpoint node j in the adjacent cluster, to disconnect from that cluster and to join C_u. If joining C_u can increase the value of the clustering then j will accept the proposal, otherwise it will reject it and no change in the clustering will take place. In particular, a detailed description of the local clustering algorithm is given in Algorithm 3. The algorithm runs at most for T_{stop} iterations, but it can easily changed so that it stops after a given number of consecutive iterations without any change of the clustering.

Algorithm 3 presents some interesting features. In fact, at every iteration, only two clusters are involved in the algorithm, while the others are idle. It is then simple to distribute the algorithm among the different clusters that can work asynchronously and in parallel as follows: at any time an inactive cluster can wake up and can propose to a node from another inactive cluster to join it, both clusters will become active until acceptance or rejection of the proposal. Several matching clusters can be active at the same time and the computations is distributed in a parallel way. Finally, being that the algorithm randomized, it is possible to run it

multiple times and then select the best solution across all the different runs.

Moreover, at every iteration, a cluster can increase by maximum one node. The complexity of the algorithm originates from calculating the function f which in its turn depends on the number of nodes in the cluster. So depending on the available computational power, we can restrict the maximum number of nodes in a cluster. For example, if the calculation of the spectral gap is affordable for graphs with only few hundred nodes, then clusters reaching this limit will stop initiating the algorithm and stop proposing to other nodes to join.

In addition, the local clustering algorithm performs well on clique-like graphs. The following simple lemma will prepare the result:

Lemma 14. *Let $g : B \to \mathbb{R}$ be a scalar strongly convex function, then for any x and y such that $x, x+1, y, y-1 \in B$ and $x \geq y$, we have:*

$$g(x) + g(y) < g(x+1) + g(y-1).$$

Proof. Let $h(x) = g(x+1) - g(x)$, since g is strongly convex, then $g'(x)$ is strictly increasing, so

$$\begin{aligned} & x+1 > x, \\ \Rightarrow\ & g'(x+1) > g'(x), \\ \Rightarrow\ & h'(x) > 0, \text{ so } h(x) \text{ is strictly increasing,} \end{aligned}$$

and adding that $x \geq y$ we can write:

$$\begin{aligned} & x > y-1, \\ \Rightarrow\ & h(x) > h(y-1), \\ \Rightarrow\ & g(x+1) - g(x) > g(y) - g(y-1), \end{aligned}$$

and the lemma follows. □

Proposition 17. *The local clustering Algorithm 3 calculates the optimal clustering for a clique graph K_n in a finite number of iterations almost surely if T_{stop} is large enough, i.e., Algorithm 3 on K_n outputs a single cluster $\mathcal{C}_G = \{V\}$.*

Proof. First note that the optimal clustering on a clique K_n is $\mathcal{C}_G^* = \{V\}$ since $f(\mathcal{C}_G = \{V\}) = 2m$ that is an upper bound on f^*. It remains to prove that the local algorithm terminates with one cluster of all nodes. Let C_u be any cluster in this graph, and let $n_u = |C_u|$ be its number of vertices, so $s_{C_u} = 1$ since the subgraph induced by C_u is also a clique, and $\rho_{C_u} = \frac{n_u - 1}{n-1}$ since the matrix $D_G(C_u)^{-1} A_G(C_u)$ has dimensions $n_u \times n_u$ and any of its elements has the value $\frac{1}{n-1}$ except the diagonal

elements that are equal to 0, therefore

$$\begin{aligned} f(C_u) &= \left(\sum_{i \in C_u} \frac{d_i^{in}}{1 + d_i^{out}} \right) s_{C_u} \rho_{C_u} \\ &= \left(\sum_{i \in C_u} \frac{n_u - 1}{1 + n - n_u} \right) \times 1 \times \frac{n_u - 1}{n - 1} \\ &= \frac{n_u (n_u - 1)^2}{(n-1)(n - n_u + 1)}, \end{aligned}$$

and it depends only on the size of the cluster. Let $g(n_u) = f(C_u)$, since $g(n_u)$ is strongly convex in n_u when $n_u \in [1, n]$, then according to the algorithm and due to Lemma 1, any node j (that belongs to the cluster C_v) receiving a proposal from a cluster C_u will accept this proposal if $|C_u| \geq |C_v|$ and will reject otherwise due to the following equation,

$$\begin{aligned} f(C_u \cup \{j\}) + f(C_v \backslash \{j\}) &= g(|C_u| + 1) + g(|C_v| - 1) && (6.3) \\ &> g(|C_u|) + g(|C_v|) && (6.4) \\ &= f(C_u) + f(C_v). && (6.5) \end{aligned}$$

The transition from (6.3) to (6.4) is due to Lemma 1. Therefore, any proposal from the cluster with largest number of vertices to other nodes is accepted (let C_{max}^k be the cluster with maximum number of vertices at iteration k), $|C_{max}^k|$ cannot decrease while it can increase by one with a probability larger than $1/n$. The algorithm terminates when $|C_{max}^k| = n$, so with probability 1 there is an iteration K such that all the nodes form a single cluster and the algorithm terminates. It is easy to check that $\mathbb{E}(K) \leq n^2$. □

6.5 Numerical Examples

In this part, we study the performance of our local clustering algorithm. We consider real world networks whose ground truth is known. We apply our algorithm (we perform multiple independent runs of the Algorithm 3 and select the best local maximum) on these networks and compare the algorithm's results with actual clustering. The results are shown using the graph visualization platform Gephi [BHJ09]. We also compare our results with the built-in modularity clustering algorithm [BGLL08] in Gephi. Our first example is the Zachary's Karate Club [Zac77], it is a social network of friendships between 34 members of a karate club at a US university in the 70s. Fig. 6.1 shows the partition of the karate club.

We apply our local clustering algorithm to the karate club network and the results are given in Fig. 6.2. Starting from 34 different clusters as initial input (every node is considered a cluster) and based on the connection and the spectral gap of the clusters, our algorithm identifies 3 clusters (one more than the ground truth). Moreover the two nodes 31 and 9 are not assigned to the correct cluster. Notice

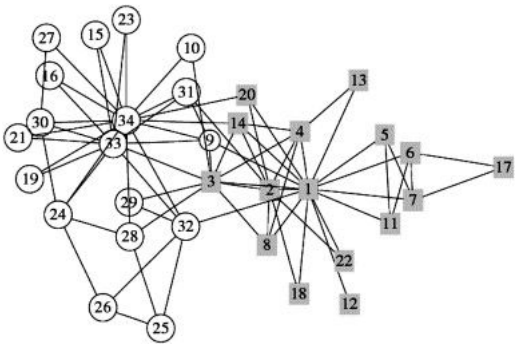

Figure 6.1: The network of social relationship between the members of the Karate Club. After a split, the members represented by a square belongs to one sub-club and the members represented by a circle to the other sub-club (the image is taken from [NG04]).

that this is just a local maximum for the optimization problem. For comparison, Fig. 6.3 shows the results of clustering using the modularity clustering algorithm, we see that it identifies even more clusters than our method (4) and node 10 is not correctly assigned in comparison to the ground truth.

The other example we consider is the network of American College football teams in Division I during Fall 2000 regular season [GN02]. Division I was made up by 115 teams divided in 12 conferences. A link in the graph corresponds to a game played between the two teams. Teams in the same conference are more likely to play games than teams from different conferences. Fig. 6.4 shows the teams grouped according to the conference they belong to. While most of conferences have good clustering properties (good connections inside the clusters and week connections among them), there are some conferences for which this is not true. For conference 1 for example there is only one game (one link) among its members, and the clubs have played most of their games against teams in different conferences. In those cases we expect the clustering algorithm to classify the nodes into different clusters.

We applied our local clustering algorithm to this network. The results are shown in Fig. 6.5. The local clustering algorithm gives 14 clusters, and we see that the algorithm was able to find correctly most of the clusters. In particular, the difference between the ground truth and the spectral gap clustering is as follows: cluster 7 was divided into two clusters, cluster 12 was also divided into two clusters. Even though conference 1 is very difficult to identify, our algorithm clustered together the only two connected nodes and clustered the disconnected nodes into different clusters. In total there are only 6 nodes that are not well clustered (out of the 115 nodes).[4]

We also present the results of clustering using the modularity algorithm of [BGLL08]. This allows us to compare the performance with other clustering al-

[4]The bad clustered nodes by Algorithm 1 in comparison to the ground truth are: 3 nodes in cluster 1, 2 nodes in cluster 9, and 1 node in cluster 5 which gives a total of 6 error nodes (without taken into consideration the split of the clusters 7 and 12).

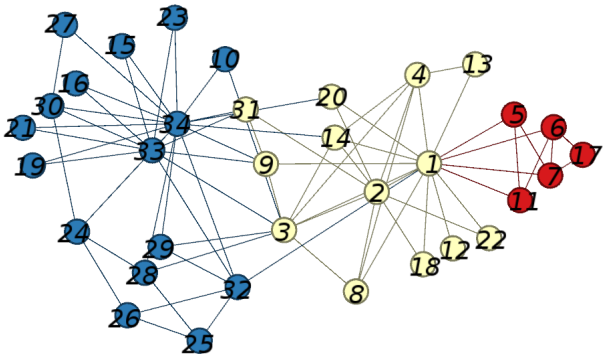

Figure 6.2: Clustering the karate club by applying Algorithm 1.

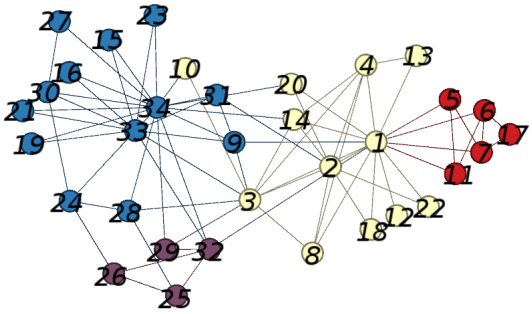

Figure 6.3: Clustering the karate club by applying modularity algorithm.

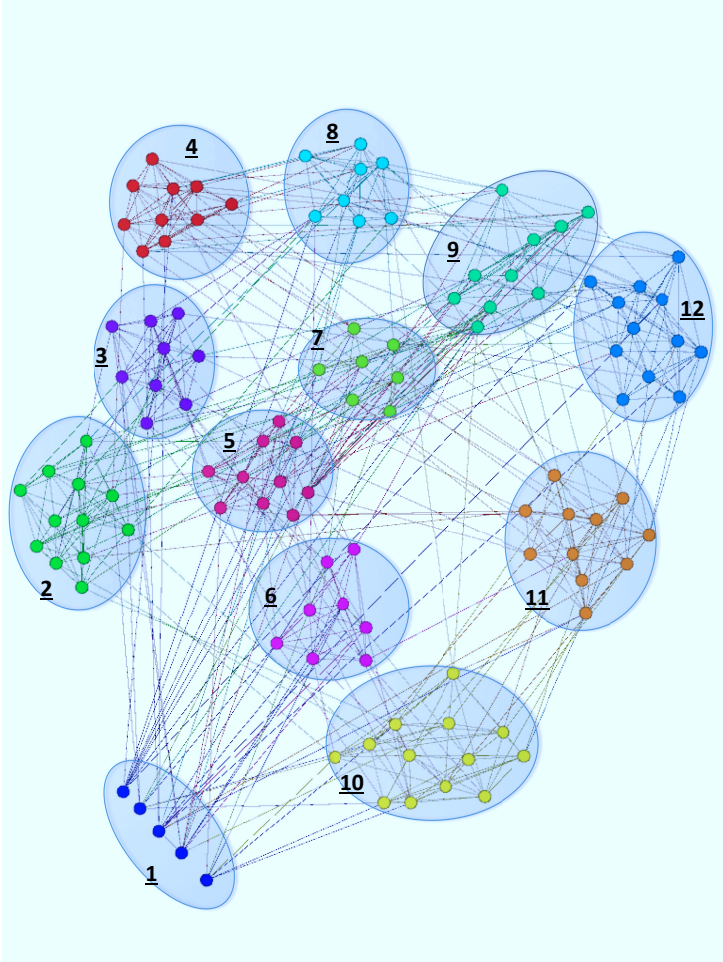

Figure 6.4: The ground truth of the conferences (clusters) in the American College football network.

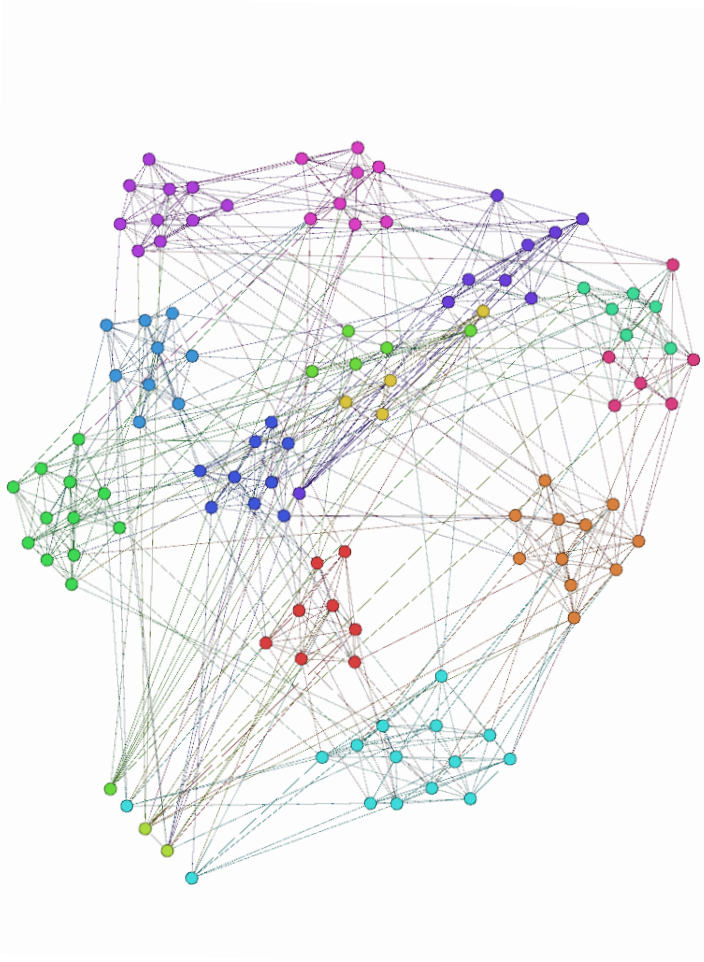

Figure 6.5: Clustering the American College football network by applying Algorithm 1. Nodes with the same color are classified as one cluster (the algorithm terminates with 14 clusters, 2 more than the ground truth).

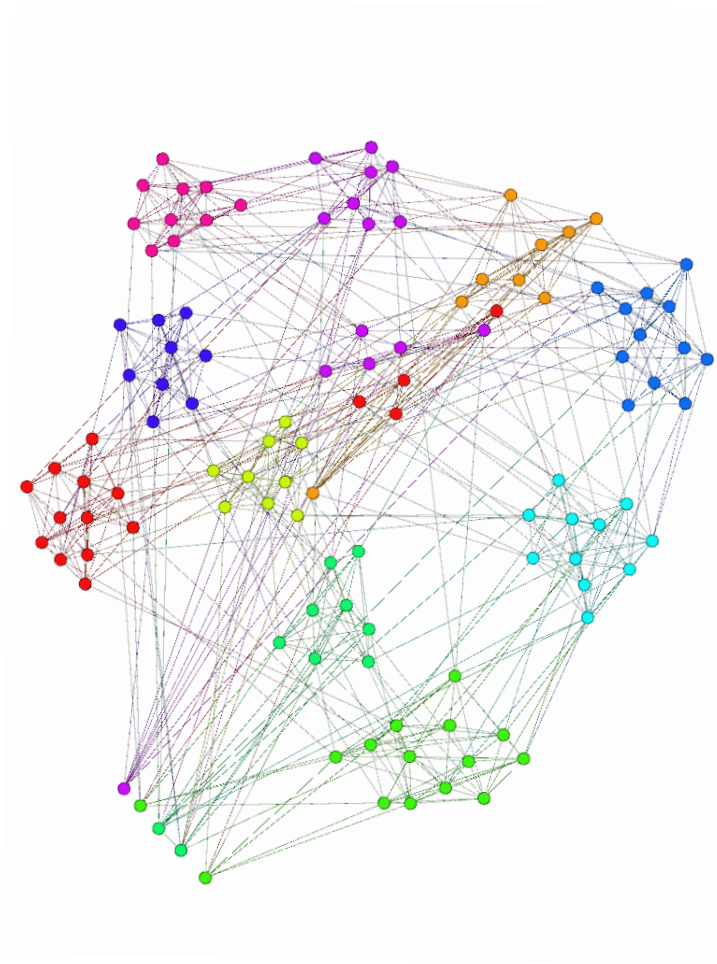

Figure 6.6: Clustering the American College football network by applying the modularity algorithm. Nodes with the same color are classified as one cluster (the algorithm terminates with 10 clusters).

gorithm and to check if the errors were due to failure of the algorithm or due to the ground truth graph structure. In the Fig. 6.6, the modularity algorithm classified the network into only 10 clusters (2 less clusters than the ground truth). Cluster 7 nodes were divided between two already existing clusters. Cluster 1 disappeared. Note that the same 6 nodes that were miss-classified by our algorithm were also here miss-classified which suggests that the errors are due to the structure but not to the algorithm.

6.6 Conclusion

In this chapter we proposed a new clustering metric based on the spectral gap of a random walk on clusters. We also proposed a randomized local clustering

algorithm that outputs a locally optimal clustering of the graph. The algorithm can be distributed in a network and clusters are iteratively updated on the basis of local communication and processing. One of the strengths of our algorithm is its ability to detect small clusters. The complexity can also be adapted to available processing capabilities.

Chapter 7

Conclusion and Perspectives

In this thesis, optimization, control, and game theoretical problems related to consensus protocols were studied and analyzed.

In Chapter 2, we have proposed an approximated solution for achieving fast consensus of the distributed linear averaging problem. This solution is obtained by minimizing the Schatten p-norm of the weight matrix where p is a parameter in the proposed problem. This solution has the advantage of being suitable for a distributed implementation with guaranteed error bounds compared to the optimal algorithm. By tuning the parameter p in our proposed minimization, we can simply trade-off the quality of the solution (i.e., the speed of convergence) for communication/computation requirements (in terms of number of messages exchanged and volume of data processed). Simulations on random graphs show that it also outperform other known distributed weight selection techniques even with limited communication overhead.

As a future work, in this direction, it would be interesting to study this approximation on dynamic graph topology. On these networks, the distributed implementation of Schatten p-norm minimization can automatically adapt to the changes in the topology contrary to other globally and centralized solutions optimized for static graphs. The good performance of Schatten p-norm on graph optimization problems opens the way to revisiting and applying this norm to other norm minimization applications in machine learning, resource allocation, network tomography, etc.

In Chapter 3, we have studied a finite-horizon discrete-time optimal control problem for a network designer to achieve faster consensus given the network structure and the initial nodes' values. The optimal control is obtained using gradient methods. We have also provided sufficient conditions for reaching consensus in one stage. Moreover, we have studied the saddle-point equilibrium (SPE) of the consensus problem in the presence of an adversary, and found that an SPE does not exist in pure strategies. If the adversary plays first, then the network designer can adjust the weights in such a way that the effect of the adversary can be removed. However, if the network designer plays first, the adversary can affect the optimal value and would drive the system away from consensus. Nevertheless, an SPE exists in mixed strategies, where the adversary selects the noise using a randomized strategy, whereas the network designer's strategy is still pure.

As future work, distributed implementation of the optimal control would be an important direction because of the distributed nature of the average consensus protocols. So far, the adversary has access to initial values; it would be interesting to remove the dependence of the strategies on the initial values. Moreover, considering

a broader class of adversaries (as malicious and misbehaving nodes, or adversaries that break links) is also one of our future research directions.

In Chapter 4, we have studied the performance of deterministic distributed averaging protocols subject to communication quantization. We have shown that quantization due to links can force quantization on the state. Depending on initial conditions, the system converges in finite time to either a quantized consensus, or the nodes' values are entering into a cyclic behavior oscillating around the average. We have derived tight error bounds for the cycle size.

As future work, it will be interesting to quantify the length of a period of the cycle of the system. We also plan to extend our results to the cases when the graph may change over time and the agents may update their variables in an asynchronous manner. Although cyclic behavior of the system will generally not occur for the above cases, simulations show that the quantized consensus can still be achieved. The analysis tools presented in this thesis are promising for these more complicated and challenging cases.

In Chapter 5, we proposed a locally-based algorithm for reducing the communication overhead due of the messages between agents running consensus protocols. We proved that this algorithm is converging to a consensus at most η from the true average. The algorithm can be applied on dynamic graphs and is also robust and adaptive to errors caused by a node suddenly changing its estimate.

A new notion, *asymptotic termination*, was introduced which is a weaker condition than finite time termination. So as future work, it would be interesting to investigate more this notion specially to domains where finite time termination algorithms do not exist.

In Chapter 6, we proposed a scoring clustering metric based on the random walk's properties to evaluate the quality of a cluster on nodes. We also proposed a randomized algorithm that identifies a locally optimal clustering of the graph according to the metric defined. The algorithm is intrinsically distributed and asynchronous.

As future work, it would be interesting to investigate more the proposed metric because it shows promising results on small/medium sized networks.

Appendix A

Open Research Direction: Averaging on Networks with Dynamic Nodes

Contents

A.1 Introduction

The purpose of this appendix chapter is to introduce a novel research direction in consensus protocols. In dynamic networks, the topology of the nodes in a network or links connecting these nodes change with time. This can be due to mobility, link failure, or nodes failure. We are interested in this chapter to study consensus on dynamic networks. Most of the work on consensus in dynamic network settings consider a fixed number of nodes that are trying to reach agreement in the presence of either mobility or non-robust links (so only the links are dynamic) [OSM04]. Average consensus on these networks is modeled following a random adjacency matrix $A^{n\times n}(k)$ where n is the number of nodes and is fixed while the elements of this matrix are random (being for example i.i.d. at every iteration k). The study of consensus on that model is reduced to studying the convergence of the backward product of random matrices. Some papers give sufficient conditions on the weight matrices at every time iteration that guarantee convergence, others use coefficient of ergodicity as a tool to show the convergence of their system to consensus [TSJ08]. However, little study has been made on networks with dynamic number of nodes. In the latter case, the dimensions of the adjacency (and weight) matrices can be unbounded, and thus the traditional tools for studying the consensus are not applicable. We refer in this report to this type of networks: nodes arrive and leave as in

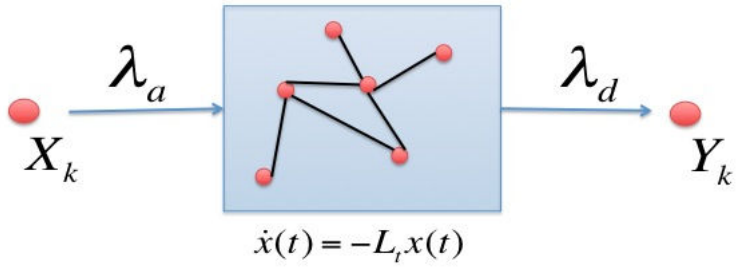

Figure A.1: The network model.

a queuing system. We would like to study the effect of averaging in these networks, and to see if the nodes could actually reach consensus.

A.2 Model

Networks with dynamic nodes are characterized by nodes' arrival and nodes' departure (see Fig. A.1). Each node arrives with a random value X_k (where k is the number of arrival and each node is labeled by its arrival number). We suppose that there is initially a node having a label 0 in the system that does not leave (it arrives at time $t = 0$ and does not depart). Let $X_0, X_1, X_2, \ldots$ be pairwise independent random variables following some distribution. Let A_k be the arrival time of node k and D_k its departure time. At any instance t, nodes within the system are connected to each other by some network topology. Let $V(t)$ be the set of all nodes in the system at time instant t, i.e., $V(t) = \{k \mid A_k \leq t < D_k\}$. Let $Y_k(t)$ be the value of node k that changes with time depending on its history of connection with other nodes, for example it should be clear that $Y_k(t)$ has a constant value before arrival ($Y_k(t) = X_k$ for $t < A_k$) and after departure ($Y_k(t) = Y_k(D_k)$ for $t \geq D_k$) but this value changes during the time it spends in the system because of interaction with other nodes. The nodes perform a continuous time averaging:

$$\dot{\mathbf{x}}(t) = -L_t \mathbf{x}(t), \tag{A.1}$$

where $\mathbf{x}(t)$ is the state vector of the nodes present in the system at time t, and $x_i(t) = Y_{\gamma(i)}(t)$ where $\gamma(i)$ is the arrival number of the i-th oldest node in the system (notice that $x_1(t) = Y_0(t)$ is always true for the node labeled 0 because it is the oldest node in system and does not depart). L_t is the Laplacian of the graph at time t, and $\dot{\mathbf{x}}(t) = \frac{\partial \mathbf{x}(t)}{\partial t}$. We call this model a *consensus queue* due to its similarity to queuing systems. If the inter-arrival and inter-leave times are exponentially distributed random variables with a FIFO discipline (the nodes first to come are the first to leave except for node 0), then the system is $M/M/1$ consensus queue with arrival rate λ_a and departure rate λ_d.

The Laplacian of the graph L_t is used to give a general model for different graph topologies. As start up results, we will give some simplifications in the next section:

1) we only consider two graph topologies, the complete graph and the tree, 2) the averaging is faster than the dynamics of the queue (i.e., equation (A.1) converges before the arrival or the departure of a new node).

The model described here is interesting because it can be applied to different and diverse applications. Queuing consensus can be for example a model for human interactions and their behavior. Consider a system where people arrive at an open market and products' prices are not fixed. Each customer has an initial estimation of the price of the product that varies depending on the interaction process with other customers in the system. You can also consider this model as a representative of a wireless sensor network where sensors monitor some environmental measurement (as temperature or pressure) where nodes can fail according to a Poisson process and new nodes are added to the network. We are interested then by the average in the system, mainly by:

$$Z(t) = \frac{1}{N(t)} \sum_{i=1}^{N(t)} x_i(t), \tag{A.2}$$

where $N(t) = |V(t)|$ is the number of nodes that are present in the system at time t and $x_i(t)$ are their estimates. We note that $Z(t)$ is not a continuous process in general because with every arrival (or departure) with an estimate X_k (or Y_k) different from $Z(t)$, the process jumps to a different value.

A.3 Simple Network Topologies

There are two sources of randomness in this model, the first one is the input estimate X that follows some distribution, and the other one is the queuing system with random arrivals and departures. In the following sections, we will characterize the average in the system and the output process by considering several simplifications:

1. Complete Graph: the network is a full mesh network (all nodes in the system are connected to each others) and once a node enters a system, all nodes will have the average of nodes presented (instantaneous averaging), i.e., $x_i(t) = Z(t)$ for all i in the system.

2. Directed Tree: nodes arriving can only connect to one node chosen uniformly at random in the system, and their estimate changes only once till they leave the network depending on the chosen node's estimate and it's distance from the root.

A.3.1 Complete Graph

Let Z_k be the value of $Z(t)$ just after the k-th arrival and before the $k+1$-th arrival. Then Z_k can be written as a weighted average of the nodes in the network, i.e., $Z_k = \sum_{i=0}^{k} w_i X_i$, where w_i is the weight given to the value of node i and it is a random variable depending on the stochastic arrivals and departures. It is important to study these weights to see how the system preserves the history of old values. To

do this we take two extreme cases. The first case of no departures (if $\lambda_d = 0$), then we have

$$w_i = \frac{1}{k+1} \text{ for } i = 0, \ldots, k. \tag{A.3}$$

The second case of very fast departures ($\lambda_d >> \lambda_a$), then we suppose each node that arrives, averages with node 0 and then leaves the system, so

$$w_i = \begin{cases} (\frac{1}{2})^{k+1-i} & \text{for } i = 1, \ldots, k, \\ (\frac{1}{2})^k & \text{for } i = 0. \end{cases} \tag{A.4}$$

The results are interesting as they show that if the system's departure rate is fast, then the weights for old values decrease exponentially, but if the departure is slow then the weights for old values decrease linearly in k.

A.3.2 Directed Tree

For this topology, we assume that there are no departures from the system. Each node arrives, connects uniformly at random to one node (we call it its parent) in the network. Let L be the distance from a newly connected node to the root (node 0). We suppose that each node i arrives at level L, connects to its parent j, and then averages as follows:

$$Y_i(t) = \frac{1}{L} X_i + \frac{L-1}{L} Y_j(t) \text{ for } t \geq A_i.$$

Notice that $Y_i(t) = \frac{1}{L} \sum_{s \in P} X_s$ is a constant for $t > A_i$ where P is the set of nodes on the path from the root to i. Since L converges to $\log n$ in probability as $n \to \infty$ [Dev88], we conclude that if $X_1, X_2, \ldots$ are i.i.d random variables of mean μ, then the value $Y_i(t)$ for $t > A_i$ converges in probability to μ as $i \to \infty$.

A.4 Conclusion

In this appendix we have presented, as an open problem, a novel model for averaging in networks with dynamic nodes. As start up results, we studied the average process in simple network settings: the complete graph and the tree topology. The results for the complete graph show that if the system's departure rate is fast, then the weights for old values decrease exponentially, but if the departure is slow then the weights for old values decrease linearly in k. It would then be interesting to characterize the decrease in the average weight of the history as function of the performance parameter of the queue (i.e., as function of $\rho = \lambda_a/\lambda_d$). For directed trees, we have shown that the value of a new arrival after averaging with its parent in the network converges in probability to its mean value. It would then be interesting to check if stronger convergence results hold. Therefore, the study of the general model (e.g., the study of the expected value of the average process $Z(t)$ given the initial values) is an open problem.

Appendix B

Présentation des Travaux de Thèse en Francais

Contents

B.1 Introduction

Depuis bien longtemps nous savons que nous sommes interconnectés, et pas seulement les uns avec les autres, mais aussi avec l'environnement réseau qui nous entoure. La plupart des réseaux actuels sont fortement interconnectés. L'Internet, le web, les réseaux de communication, les réseaux de capteurs sans fil, les réseaux intelligents, et les réseaux sociaux ne sont que quelques exemples d'environnements interconnectés. La caractéristique commune intéressante de ces réseaux est qu'ils peuvent être composés de nombreux petits sous-systèmes qui prennent les décisions locales (basée uniquement sur les règles d'interaction voisin). Ces décisions locales peuvent avoir un impact crucial sur l'ensemble du réseau. Par exemple, un virus qui se répand à partir d'un ordinateur infecté, peut faire de graves dommages dans le réseau. Un partage de vidéo par un utilisateur bien connecté aux réseaux sociaux peut faire le continu multimédia viral et atteint une grande partie de la population.

En général, un réseau est formé de noeuds (ou agents) et des liens de communication qui permettent à ces nœuds de partager des informations et des ressources.

Dans ce manuscrit on définit un“agent” comme un automate (peut-être une machine avec un nombre infinis d'états) programmé pour exécuter des algorithmes en fonction de la dynamique des interactions bien définies. Ces dynamiques changent les états des agents (et donc l'état du système), et en fonction des décisions locales, ces états peuvent converger ou pas. Les décisions locales qui causent les états des agents de converger vers un état commun, sont appelées *les protocoles de consensus*. Dans ce travail, nous ne considérons que les systèmes en temps discret, bien que les algorithmes étudiés peut être utilisé pour étudier les systèmes en temps continu aussi.

Les protocoles de consensus peuvent être appliqués dans divers contextes de réseaux (comme ceux mentionnés plus haut) où les interactions entre voisins sont possibles. En effet, ces protocoles se situent à l'intersection de différents domaines de recherche comme la théorie des systèmes, des modèles et systèmes de calculs, et la théorie des graphes. La théorie des systèmes est l'étude transdisciplinaire de l'organisation abstraite des phénomènes, sans être spécifique à un type précis d'objets, aux leurs propriétés exactes, ou à la description qualitative de leurs règles d'interaction dans l'environnement sous-jacent. Cette abstraction dans les protocoles de consensus est donnée par la modélisation du réseau par un graphe de sommets (les agents) reliés par des liens (s'ils communiquent), puis en exécutant des algorithmes de consensus au dessus de cela.

Par ailleurs, nous étudions des problèmes d'optimisation concernés par les protocoles de consensus en temps discret sur les réseaux, comme paramètre de réglage pour augmenter la vitesse de convergence, la mise en œuvre distribuée de solutions d'optimisation globale, et la réduction de la surcharge de communication (Chapitres 2 et 5). Nous proposons également un cadre conceptuel d'analyse des jeux qui prend en compte un adversaire dans le réseau en essayant de perturber le canal de communication (Chapitre 3). Nous analysons des algorithmes de consensus en présence de contraintes de communication comme la quantification (Chapitre 4). Nous abordons le problème des regroupements (clustering) dans un réseau en proposant une mesure de similarité basé sur la vitesse de convergence des protocoles de consensus et les propriétés de trou spectral des marches aléatoires (Chapitre 6).

Les motivations principales de cette thèse sont les trois applications suivantes où les protocoles de consensus sont fondamentaux dans leur conception:

- Optimisation et contrôle distribué
- Monitoring environnemental
- Système multi-agents

Ainsi, la contribution de cette thèse est d'enrichir les connaissances dans la recherche sur les protocoles de consensus en général et à ces applications en particulier.

B.1.1 Optimisation et Contrôle Distribué

Il y a un nombre important de recherches sur l'optimisation distribuée dans les réseaux. De nouvelles techniques rapides [WOJ13, GJS11] ont été proposées pour la dualisation lagrangienne des problèmes séparables. Celle-ci est bien connue dans la communauté du réseau depuis le travail séminal de Kelly sur TCP [KMT98]. D'autres travaux dans [NO09, JKJJ08] combinent un protocole de consensus, qui est utilisé pour distribuer les calculs entre les agents, et une méthode de sous-gradient pour la minimisation d'un objectif local. Une approche différente repose sur une exploration aléatoire intelligente de l'espace de solution possible, par exemple, en utilisant des algorithmes génétiques [ANC+10] ou l'échantillonneur de Gibbs [KBC+07]. En effet, l'optimisation distribuée par des protocoles de consensus dans la communauté de contrôle remonte aux années 80 grâce au travail de D. P. Bertsekas et J. N. Tsitsiklis sur la prise de décision décentralisée et le calcul parallèle [BT89].

Les problèmes de consensus ont aussi une relation étroite avec l'algorithme PageRank utilisé par le moteur de recherche Google pour classer les pages web pour les requêtes de recherche [BP98]. Comme le nombre de sites jusqu'à présent est plus de 1 milliard,[1] le PageRank nécessite le calcul d'un vecteur propre correspondant à la plus grande valeur propre de très grande matrice. Par conséquent, l'utilisation de l'information globale n'est pas possible dans ce cas, et les implémentations parallèles et distribuées sont obligatoires [LM06, ALNO07]. Une possibilité est d'utiliser des algorithmes de consensus [IT10, ANP07]. Le problème de PageRank a suscité récemment l'intérêt de la communauté de contrôle [IT14].

Dans les réseaux, des algorithmes de routage efficaces et une utilisation efficace des ressources sont proposés pour économiser l'énergie et accélérer le traitement. Pour les petits réseaux, il est possible pour une unité centrale d'être au courant de tous les composants du réseau et de décider comment utiliser de façon optimale une ressource. Pour les grands réseaux, l'unité centrale doit gérer une grande quantité de données, et l'optimisation centralisée peut devenir impossible, en particulier lorsque le réseau est dynamique [BFH13]. En effet, la configuration optimale doit être calculée à chaque fois qu'une liaison tombe en panne ou qu'un changement se produise dans le réseau. En outre, les noeuds peuvent avoir des capacités de traitement qui ne sont pas utilisées dans l'optimisation centralisée. Par conséquent, il convient d'effectuer l'optimisation distribuée basée sur le calcul locale à chaque nœud et sur l'information locale entre les voisins [Joh08]. Cette approche distribuée est intrinsèquement capable de s'adapter aux changements du réseau local.

B.1.2 Monitoring Environnemental

Les nouvelles technologies comme la robotique, le contrôle du groupe de véhicules et la surveillance de l'environnement ont besoin de réseaux de capteurs sans fil. Dans ces réseaux, un groupe de capteurs communique d'une manière ad hoc pour accomplir les tâches à faire.

[1] www.internetlivestats.com

En surveillance de l'environnement, des capteurs mesurent la température, la pression, la pollution, etc. dans leur zone de déploiement. Ces mesures peuvent être bruyants, et si le bruit est additif, de moyenne nulle et gaussien, chaque capteur de température peut avoir une mesure différente de la température de consigne. Il est bien connu qu'un bon filtre de bruit gaussien (pour atteindre le maximum de vraisemblance) est le filtre de moyenne. Par conséquent, la moyenne des valeurs des mesures initiales peut donner une estimation plus précise. Ceci est connu comme *la fusion de capteurs*. La fusion des capteurs peut être obtenue par la communication décentralisée entre les capteurs en utilisant des protocoles de consensus. En effet, la fusion de capteurs est la motivation communiquée par Boyd *et al.* pour leur article bien connu sur les protocoles de bavardage (gossiping protocols) en consensus [BGPS06].

B.1.3 Système Multi-agents

Les protocoles de consensus sont également utilisés dans les problèmes de coordination multi-agents [OSFM07]. Les agents de ces réseaux utilisent également la technologie de capteurs sans fil pour communiquer. Un groupe de robots se déplaçant en parallèle par exemple devrait s'accorder sur la direction du mouvement et de la vitesse pour éviter la collision. La difficulté pour les protocoles de consensus dans cette catégorie de problèmes n'est pas le grand nombre de robots, mais plutôt c'est une question de la dynamicité de topologie et de connectivité.

B.2 Les Contributions de la Thèse

Nous étudions dans cette thèse les problèmes d'optimisation, de contrôle, et de théorie de jeu qui se posent dans les protocoles de consensus. En particulier, nous soulevons les questions suivantes:

- Les nœuds du réseau peuvent-ils éventuellement déduire (ou apprendre) les poids optimaux de manière distribuée pour accélérer la convergence des protocoles de consensus (sans la nécessité de la connaissance globale du réseau)?
- Étant donné que le réseau peut être sensible aux attaques par un adversaire prêt de conduire le système loin de consensus, quelles stratégies devraient être utilisées par l'adversaire et le concepteur du réseau pour atteindre leurs objectifs opposés?
- Supposons que les canaux de communication entre les nœuds du réseau sont soumis à des contraintes, et les nœuds reçoivent/envoient des valeurs tronquées. Comment peut cette quantification influer la convergence du système non linéaire qui en résulte?
- Étant donné que les variables convergent asymptotiquement, les nœuds peuvent-ils savoir quand leurs variables d'état sont assez proches de la valeur

asymptotique, et donc décider d'arrêter l'execution d'algorithme basée uniquement sur une connaissance locale?

- Étant donné que les noeuds formant des groupes bien connectés dans le graphe ont une dynamique de convergence similaires, comment on peut utiliser cette observation pour déterminer ces groupes de nœuds?

Chaque chapitre de la thèse traite de l'une des questions ci-dessus.

B.2.1 Sélection de Poids dans les Protocoles de Consensus

Nous approchons le problème de la sélection optimale de poids avec un problème de minimisation de la p-norme de Schatten. Ce dernier est résolu de manière totalement distribuée grâce à une mèthode du gradient. Selon la valeur du paramètre p, nous pouvons trouver un compromis entre la qualité de la solution (c'est-à-dire la vitesse de convergence) et les coûts en termes de communication et calcul (nombre de messages échangés et volume de données traitées). Les résultats des simulations montrent que notre approche fournit une très bonne performance même avec un échange d'informations limité. La procédure d'optimisation des poids peut également se dérouler simultanement avec le protocole de consensus.

Les publications directement liées à cette contribution sont les suivantes:

- [ECNA15] M. El Chamie, G. Neglia, and K. Avrachenkov, "Distributed Weight Selection in Consensus Protocols by Schatten Norm Minimization", To appear in IEEE Transactions on Automatic Control as Technical Note, Volume 60, No. 4, April 2015.

- [ECN14] M. El Chamie and G. Neglia, "Newton's Method for Constrained Norm Minimization and Its Application to Weighted Graph Problems", In proceedings of the American Control Conference ACC 2014 (Portland, OR, United States, June 4-6), pp. 6, June 2014.

Autres publications également liés à ce sujet sont les suivantes:

- [SECN13] L. Severini, M. El Chamie, and G. Neglia, "Topology versus Link Strength for Information Dissemination in Networks", In proceedings of ALGOTEL 2013 (Pornic, Loire-Atlantique, France, May 28-31), pp. 4, May 2013.

- [AECN11] K. Avrachenkov, M. El Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks", In proceedings of IEEE International Conference on Distributed Computing in Sensor Sytems and Workshops DCOSS 2011 (Barcelona, Spain June 27-29), pp. 6, June 2011.

B.2.2 Un Adversaire dans les Protocoles de Consensus

Nous proposons un cadre conceptuel d'analyse des jeux d'adversaire qui peut ajouter du bruit aux poids utilisés par l'algorithme de consensus de moyenne afin d'éloigner

le système de consensus. Nous donnons les stratégies optimales pour les joueurs (l'adversaire et le concepteur du réseau) dans ce jeu et nous montrons qu'un point-selle (saddle-point equilibrium) existe dans les stratégies mixtes.

La publication relative à cette contribution est la suivante:

- [ECB14] M. El Chamie and T. Başar, "Optimal Strategies for Dynamic Weight Selection in Consensus Protocols in the Presence of an Adversary", Accepted to the 53rd IEEE Conference on Decision and Control CDC 2014 (Los Angeles, California, Dec. 15-17), pp. 6, Dec. 2014.

B.2.3 Conception et Analyse d'Algorithmes Distribués de Moyennage avec Valeurs Échangées Discrétisées

L'objectif de ce travail est d'étudier les performances d'une sous-classe d'algorithmes déterministes de calcul de la moyenne distribuée, où l'échange d'informations entre les nœuds voisins est soumis à la quantification uniforme. Avec une telle quantification, la moyenne précise ne peut être atteinte (sauf dans des cas exceptionnels), mais une valeur proche d'elle peut être atteinte. Cette valeur est appelée consensus quantifié. Nous montrons que, dans un temps fini, soit tous les n agents parviennent à un consensus quantifié où la valeur de consensus est le plus grand entier qui n'est pas supérieur à la moyenne de leurs valeurs initiales; ou soit tous les n agents cyclent dans un petit voisinage autour de la moyenne, en fonction des conditions initiales. Dans ce dernier cas, il est démontré que le voisinage puisse être rendu arbitrairement faible en ajustant les paramètres de l'algorithme de manière distribuée.

La publication relative à cette contribution est la suivante:

- [ECLB14] M. El Chamie, J. Liu, and T. Başar, "Design and Analysis of Distributed Averaging with Quantized Communication", Accepted to the 53rd IEEE Conference on Decision and Control CDC 2014 (Los Angeles, California, Dec. 15-17), pp. 6, Dec. 2014.

B.2.4 La Réduction de Charge de Communication dans les Protocoles de Consensus

La convergence du consensus de moyenne est asymptotique et la mise en œuvre d'un protocole de terminaison est difficile lorsque les nœuds ne connaissent pas l'estimation globale (par exemple, le diamètre du réseau ou le nombre de nœuds). Nous nous intéressons à la réduction du taux de messages envoyés dans le réseau quand les estimations deviennent proche du consensus. Nous présentons un algorithme de consensus de moyenne totalement distribué, où les nœuds envoient plus de messages lorsque la différence entre leurs estimations est grande et moins de messages lorsque le système est à peu prés convergeant. La convergence du système est garantie d'être proche de la vraie moyenne et le coût des communications est fortement réduit.

La publication relative à cette contribution est la suivante:

- [ECNA13] M. El Chamie, G. Neglia, and K. Avrachenkov, "Reducing Communication Overhead for Average Consensus", In proceedings of IFIP Networking 2013 (Brooklyn, NY, USA, May 22-24), May 2013.

B.2.5 Regroupement

Nous proposons une mesure de similarité qui évalue la qualité d'un regroupement (clustering) des nœuds dans un réseau. Un algorithme local de clustering basé sur cette métrique est proposé.

La publication relative à cette contribution est la suivante:

- [AECN14] K. Avrachenkov, M. El Chamie, and G. Neglia, "Graph Clustering Based on Mixing Time of Random Walks", In proceedings of the IEEE International Conference on Communications ICC 2014 (Sydney, Australia, June 10-14), pp. 6, June 2014.

B.3 Conclusion

Dans cette thèse, nous avons étudié les problèmes d'optimisation, de contrôle, et de théorie de jeu qui se posent dans les protocoles de consensus.

D'abord, nous avons étudié les techniques d'optimisation pour des problèmes de sélection de poids permettant ainsi d'augmenter la vitesse de convergence de protocoles de consensus dans les réseaux. Nous avons proposé de sélectionner les poids en appliquant un algorithme d'approximation: minimisation de la norme p de Schatten de la matrice de poids. Nous avons caractérisé l'erreur induite par cette approximation et nous avons montré que l'algorithme proposé a l'avantage qu'il peut être soit résolu de façon distribuée en utilisant une méthode de gradient projeté simple ou résolu par la méthode de Newton et avec une convergence plus rapide.

Ensuite, nous avons proposé un cadre conceptuel d'analyse des jeux d'adversaire qui peut ajouter du bruit aux poids utilisés par l'algorithme de consensus de moyenne afin d'éloigner le système de consensus. Nous avons donné les stratégies optimales pour les joueurs (l'adversaire et le concepteur du réseau) dans ce jeu et nous montrons qu'un point-selle (saddle-point equilibrium) existe dans les stratégies mixtes.

Nous avons analysé également la performance des algorithmes de consensus de moyenne où les informations échangées entre les agents voisins sont soumises à la quantification uniforme déterministe (les valeurs réelles envoyées par les nœuds de leurs voisins sont tronquées). En utilisant la notion de stabilité au sens de Lyapunov, nous avons caractérisé les propriétés de convergence du système quantifié non linéaire résultant.

Le problème de la terminaison des protocoles de consensus s'avère difficile dans le cadre distribué. Nous avons proposé un algorithme distribué pour la terminaison des protocoles de consensus. L'algorithme réduit la charge de communication tout en garantissant la convergence vers un consensus. Enfin, nous avons proposé

une mesure de similarité qui évalue la qualité d'un regroupement (clustering) des nœuds dans un réseau. Un algorithme local de clustering basé sur cette métrique est proposé.

Bibliography

[AAD+04] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta, *Computation in networks of passively mobile finite-state sensors*, Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing (New York, NY, USA), PODC '04, ACM, 2004, pp. 290–299. (Cited in page 4.)

[AAMR93] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao, *Approximate load balancing on dynamic and asynchronous networks*, Proceedings of the 25th Annual ACM Symposium on Theory of Computing, 1993, pp. 632–641. (Cited in page 73.)

[AB10] T.C. Aysal and K.E. Barner, *Convergence of consensus models with stochastic disturbances*, IEEE Transactions on Information Theory **56** (2010), no. 8, 4101–4113. (Cited in pages 72 and 73.)

[ACFO11] D. Acemoglu, G. Como, F. Fagnani, and A. Ozdaglar, *Opinion fluctuations and persistent disagreement in social networks*, 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC 2011), Dec. 2011, pp. 2347 –2352. (Cited in pages 51 and 57.)

[ACL06] R. Andersen, Fan Chung, and K. Lang, *Local graph partitioning using pagerank vectors*, 47th Annual IEEE Symposium on Foundations of Computer Science, 2006 (FOCS '06), 2006, pp. 475–486. (Cited in page 124.)

[ACR07] T. C. Aysal, M. Coates, and M. Rabbat, *Distributed average consensus using probabilistic quantization*, Proceedings of the 14th IEEE/SP Workshop on Statistical Signal Processing, 2007, pp. 640–644. (Cited in page 72.)

[AECN11] K. Avrachenkov, M. El Chamie, and G. Neglia, *A local average consensus algorithm for wireless sensor networks*, 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), June 2011, pp. 1–6. (Cited in pages 11, 20 and 149.)

[AECN14] Konstantin Avrachenkov, Mahmoud El Chamie, and Giovanni Neglia, *Graph clustering based on mixing time of random walks*, IEEE International Conference on Communications ICC 2014 (Sydney, Australia, June 10-14), Jun 2014, p. 6. (Cited in pages 13 and 151.)

[AFSU07] Yonatan Amit, Michael Fink, Nathan Srebro, and Shimon Ullman, *Uncovering shared structures in multiclass classification*, Proceedings of the 24th international conference on Machine learning (New York, NY, USA), ICML '07, ACM, 2007, pp. 17–24. (Cited in page 52.)

[AGMZ11] Hélio Almeida, Dorgival Guedes, Wagner Meira, and Mohammed J. Zaki, *Is there a best quality metric for graph clusters?*, Proceedings of the 2011 ECML PKDD'11 (Berlin, Heidelberg), Springer-Verlag, 2011, pp. 44–59. (Cited in page 124.)

[ALNO07] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, *Monte carlo methods in pagerank computation: When one iteration is sufficient*, SIAM Journal on Numerical Analysis **45** (2007), no. 2, 890–904. (Cited in pages 3 and 147.)

[AMP10] Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil, *On spectral learning*, J. Mach. Learn. Res. **11** (2010), 935–953. (Cited in page 53.)

[AMPY07] Andreas Argyriou, Charles A. Micchelli, Massimiliano Pontil, and Yiming Ying, *A spectral regularization framework for multi-task structure learning*, In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, Advances in Neural Information Processing Systems 20, MIT Press, 2007. (Cited in page 52.)

[ANC+10] Sara Alouf, Giovanni Neglia, Iacopo Carreras, Daniele Miorandi, and Álvaro Fialho, *Fitting Genetic Algorithms to Distributed On-line Evolution of Network Protocols*, Elsevier Computer Networks **54** (2010), no. 18, 3402–3420. (Cited in pages 3 and 147.)

[ANP07] Konstantin Avrachenkov, Danil Nemirovsky, and Kim Son Pham, *A survey on distributed approaches to graph based reputation measures*, Proceedings of the 2Nd International Conference on Performance Evaluation Methodologies and Tools (ICST, Brussels, Belgium, Belgium), ValueTools '07, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007, pp. 82:1–82:9. (Cited in pages 3 and 147.)

[ARS12] H. Attouch, P. Redont, and B.F. Svaiter, *Global convergence of a closed-loop regularized newton method for solving monotone inclusions in hilbert spaces*, Journal of Optimization Theory and Applications (2012), 1–27 (English). (Cited in page 25.)

[BA98] T. Balch and R.C. Arkin, *Behavior-based formation control for multirobot teams*, IEEE Transactions on Robotics and Automation **14** (1998), no. 6, 926–939. (Cited in page 4.)

[BABJ12] Walid Ben-Ameur, Pascal Bianchi, and Jérémie Jakubowicz, *Robust average consensus using total variation gossip algorithm*, 6th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2012), IEEE, Nov. 2012. (Cited in pages 51 and 57.)

[Bén09] Florence Bénézit, *Distributed average consensus for wireless sensor networks*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne (EPFL), November 2009. (Cited in page 7.)

[Ber05] D.S. Bernstein, *Matrix mathematics: theory, facts, and formulas*, Princeton University Press, 2005. (Cited in pages 8, 20 and 54.)

[BFH13] P. Bianchi, G. Fort, and W. Hachem, *Performance of a distributed stochastic approximation algorithm*, IEEE Transactions on Information Theory **59** (2013), no. 11, 7405–7418. (Cited in pages 3 and 147.)

[BGLL08] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, *Fast unfolding of communities in large networks*, Journal of Statistical Mechanics: Theory and Experiment **2008** (2008), no. 10, P10008. (Cited in pages 131 and 132.)

[BGPS06] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah, *Randomized gossip algorithms*, IEEE/ACM Trans. Netw. **14** (2006), no. SI, 2508–2530. (Cited in pages 4, 19, 47, 73 and 148.)

[BHJ09] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy, *Gephi: An open source software for exploring and manipulating networks*, International AAAI Conference on Weblogs and Social Media, 2009. (Cited in page 131.)

[BNO03] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex Analysis and Optimization*, Athena Scientific, 2003. (Cited in pages 32 and 34.)

[BO99] Tamer Başar and Geert Jan Olsder, *Dynamic noncooperative game theory*, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 1999. (Cited in pages 65 and 67.)

[BP98] Sergey Brin and Lawrence Page, *The anatomy of a large-scale hypertextual web search engine*, Comput. Netw. ISDN Syst. **30** (1998), no. 1-7, 107–117. (Cited in pages 3 and 147.)

[Bro97] R. W. Brockett, *Cycles that effect change*, Motion, Control and Geometry: A Science and Technology Symposium, Washington, D.C., 1997. National Academy of Sciences Press., 1997. (Cited in page 95.)

[BT89] Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989. (Cited in pages 3 and 147.)

[BTV09] F. Bénézit, P. Thiran, and M. Vetterli, *Interval consensus: from quantized gossip to voting*, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2009, pp. 3661–3664. (Cited in pages 72 and 73.)

[BTV11] ———, *The distributed multiple voting problem*, IEEE Journal of Selected Topics in Signal Processing **5** (2011), no. 4, 791–804. (Cited in pages 72 and 73.)

[BV04] Stephen Boyd and Lieven Vandenberghe, *Convex Optimization*, Cambridge University Press, March 2004. (Cited in pages 20, 25, 26 and 55.)

[CDH+05] Guantao Chen, George Davis, Frank Hall, Zhongshan Li, Kinnari Patel, and Michael Stewart, *An interlacing result on normalized laplacians*, SIAM J. Discret. Math. **18** (2005), no. 2, 353–361. (Cited in page 126.)

[CFFZ10] R. Carli, P. Frasca, F. Fagnani, and S. Zampieri, *Gossip consensus algorithms via quantized communication*, Automatica **46** (2010), 70–80. (Cited in pages 73, 74 and 75.)

[CFSZ08] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri, *Communication constraints in coordinated consensus problems*, Automatica **44** (2008), no. 3, 671–684. (Cited in page 72.)

[Cha11] Mahmoud El Chamie, *Average Consensus on Large-Scale Networks*, Master's thesis, University of Nice Sophia Antipolis, France, 2011. (Cited in page 20.)

[CI11] K. Cai and H. Ishii, *Quantized consensus and averaging on gossip digraphs*, IEEE Transactions on Automatic Control **56** (2011), no. 9, 2087–2100. (Cited in page 73.)

[CM09] A. Censi and R. M. Murray, *Real-valued average consensus over noisy quantized channels*, Proceedings of the 2009 American Control Conference, 2009, pp. 4361–4366. (Cited in page 72.)

[CYRC13] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen, *An overview of recent progress in the study of distributed multi-agent coordination*, IEEE Transactions on Industrial Informatics **9** (2013), no. 1, 427–438. (Cited in page 72.)

[Dev88] L. Devroye, *Applications of the theory of records in the study of random trees*, Acta Inf. **26** (1988), no. 1-2, 123–130. (Cited in page 144.)

[DRL11] Ali Daher, Michael Rabbat, and Vincent K.N. Lau, *Local silencing rules for randomized gossip*, 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), June 2011, pp. 1–8. (Cited in page 105.)

[DS65] J. N. Darroch and E. Seneta, *On quasi-stationary distributions in absorbing discrete-time finite markov chains*, Journal of Applied Probability **2** (1965), no. 1, pp. 88–100 (English). (Cited in page 126.)

[EB13] S. R. Etesami and Tamer Başar, *Convergence time for unbiased quantized consensus*, Proceedings of 52nd IEEE Conference on Decision and Control (IEEE CDC), 2013. (Cited in pages 72 and 73.)

[ECB14] Mahmoud El Chamie and Tamer Başar, *Optimal strategies for dynamic weight selection in consensus protocols in the presence of an adversary*, 53rd IEEE Conference on Decision and Control CDC 2014 (Los Angeles, California, Dec. 15-17), Dec. 2014, p. 6. (Cited in pages 11 and 150.)

[ECLB14] Mahmoud El Chamie, Ji Liu, and Tamer Başar, *Design and analysis of distributed averaging with quantized communication*, 53rd IEEE Conference on Decision and Control CDC 2014 (Los Angeles, California, Dec. 15-17), Dec. 2014, p. 6. (Cited in pages 12 and 150.)

[ECN14] Mahmoud El Chamie and Giovanni Neglia, *Newton's method for constrained norm minimization and its application to weighted graph problems*, IEEE American Control Conference ACC 2014 (Portland, OR, United States, June 4-6), June 2014, p. 6. (Cited in pages 11 and 149.)

[ECNA13] Mahmoud El Chamie, Giovanni Neglia, and Konstantin Avrachenkov, *Reducing communication overhead for average consensus*, IFIP Networking 2013 (Brooklyn, NY, USA, May 22-24), May 2013. (Cited in pages 12 and 151.)

[ECNA15] ———, *Distributed Weight Selection in Consensus Protocols by Schatten Norm Minimization*, IEEE Transactions on Automatic Control as Technical Note **60** (2015), no. 4, To appear. (Cited in pages 11 and 149.)

[FB07] Santo Fortunato and M Barthelemy, *Resolution limit in community detection*, Proceedings of the National Academy of Sciences **104** (2007), 36–41. (Cited in page 124.)

[FCFZ09] Paolo Frasca, Ruggero Carli, Fabio Fagnani, and Sandro Zampieri, *Average consensus on networks with quantized communication*, International Journal of Robust and Nonlinear Control **19** (2009), no. 16, 1787–1816. (Cited in pages 72, 77 and 78.)

[FGGS09] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, *Decentralized Laplacian eigenvalues estimation for networked multi-agent systems*, Proc. of the 48th IEEE CDC/CCC 2009, Dec. 2009, pp. 2717 –2722. (Cited in page 19.)

[FHB01] M. Fazel, H. Hindi, and S.P. Boyd, *A rank minimization heuristic with application to minimum order system approximation*, Proceedings of the 2001 American Control Conference (ACC), vol. 6, 2001, pp. 4734–4739 vol.6. (Cited in page 53.)

[Fie73] Miroslav Fiedler, *Algebraic connectivity of graphs*, Czechoslovak Mathematical Journal **23** (1973), no. 2, 298–305 (eng). (Cited in page 5.)

[FN95] Michael K.H. Fan and Batool Nekooie, *On minimizing the largest eigenvalue of a symmetric matrix*, Linear Algebra and its Applications **214** (1995), 225–246. (Cited in page 9.)

[GB11] M. Grant and S. Boyd, *CVX: Matlab Software for Disciplined Convex Programming, version 1.21*, April 2011. (Cited in page 18.)

[GJS11] E. Ghadimi, M. Johansson, and I. Shames, *Accelerated gradient methods for networked optimization*, American Control Conference (ACC 2011), 29 2011-july 1 2011, pp. 1668 –1673. (Cited in pages 3 and 147.)

[GK98] P. Gupta and Kumar, *Critical power for asymptotic connectivity in wireless networks*, In Stochastic Analysis, Control, Optimization and Applications (1998), 547–566. (Cited in page 6.)

[GM96] B. Ghosha and S. Muthukrishnan, *"dynamic load balancing by random matchings*, Journal of Computer and System Sciences **53** (1996), no. 3, 357–370. (Cited in page 73.)

[GN02] M. Girvan and M. E. J. Newman, *Community structure in social and biological networks*, Proceedings of the National Academy of Sciences **99** (2002), no. 12, 7821–7826. (Cited in page 132.)

[HDM05] Yuko Hatano, Arindam Das, and Mehran Mesbahi, *Agreement in presence of noise: pseudogradients on random geometric networks*, 44th IEEE CDC-ECC, December 2005. (Cited in page 111.)

[Hil08] T. H. Hildebrandt, *Existence of a minimum of a quadratic function*, The American Mathematical Monthly **15** (1908), no. 3, 57–59 (English). (Cited in page 65.)

[HJOV14] Julien M. Hendrickx, Raphael M. Jungers, Alexander Olshevsky, and Guillaume Vankeerberghen, *Graph diameter, eigenvalues, and minimum-time consensus*, Automatica **50** (2014), no. 2, 635 – 640. (Cited in page 57.)

[HT11] J.M. Hendrickx and J.N. Tsitsiklis, *A new condition for convergence in continuous-time consensus seeking systems*, IEEE 50th CDC-ECC conference, Dec. 2011, pp. 5070 –5075. (Cited in page 111.)

[IK94] E. Isaacson and H.B. Keller, *Analysis of numerical methods*, Dover Books on Mathematics Series, Dover Publ., 1994. (Cited in page 55.)

[IT10] H. Ishii and R. Tempo, *Distributed randomized algorithms for the pagerank computation*, IEEE Transactions on Automatic Control **55** (2010), no. 9, 1987–2002. (Cited in pages 3 and 147.)

[IT14] H. Ishii and R. Tempo, *The pagerank problem, multiagent consensus, and web aggregation: A systems and control viewpoint*, IEEE Control Systems **34** (2014), no. 3, 34–53. (Cited in pages 3 and 147.)

[JE07] Meng Ji and M. Egerstedt, *Distributed coordination control of multiagent systems while preserving connectedness*, IEEE Transactions on Robotics **23** (2007), no. 4, 693–703. (Cited in page 4.)

[JKJJ08] B. Johansson, T. Keviczky, M. Johansson, and K.H. Johansson, *Subgradient methods and consensus algorithms for solving convex optimization problems*, 47th IEEE Conference on Decision and Control CDC 2008, Dec 2008, pp. 4185–4190. (Cited in pages 3, 32 and 147.)

[JLM03] A. Jadbabaie, Jie Lin, and A.S. Morse, *Coordination of groups of mobile autonomous agents using nearest neighbor rules*, IEEE Transactions on Automatic Control **48** (2003), no. 6, 988–1001. (Cited in page 4.)

[Joh08] Bjőrn Johansson, *On distributed optimization in networked systems*, Ph.D. thesis, Royal Institute of Technology (KTH), December 2008. (Cited in pages 3 and 147.)

[JXM10] Dusan Jakovetic, J. Xavier, and J.M.F. Moura, *Consensus in correlated random topologies: Weights for finite time horizon*, 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), March 2010, pp. 2974–2977. (Cited in page 19.)

[KBC+07] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot, *Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks*, 26th IEEE International Conference on Computer Communications (INFOCOM 2007), 2007, pp. 1451–1459. (Cited in pages 3 and 147.)

[KBS07] Akshay Kashyap, Tamer Başar, and R. Srikant, *Quantized consensus.*, Automatica **43** (2007), no. 7, 1192–1203. (Cited in pages 72 and 73.)

[KG09] Chih-Kai Ko and Xiaojie Gao, *On matrix factorization and finite-time average-consensus*, Proceedings of the 48th IEEE Conference on Decision and Control, held jointly with the 2009 28th Chinese Control Conference (CDC/CCC 2009), Dec 2009, pp. 5798–5803. (Cited in page 57.)

[KGP09] Yoonsoo Kim, Da-Wei Gu, and Ian Postlethwaite, *Spectral radius minimization for optimal average consensus and output feedback stabilization*, Automatica **45** (2009), no. 6, 1379 – 1386. (Cited in page 19.)

[KM86] Mirko Kriványek and Jaroslav Morávek, *Np-hard problems in hierarchical-tree clustering*, Acta Informatica **23** (1986), no. 3, 311–323. (Cited in page 124.)

[KM04] David Kempe and Frank McSherry, *A decentralized algorithm for spectral analysis*, Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, ACM, 2004, pp. 561–568. (Cited in page 19.)

[KM10] S. Kar and J. M. F. Moura, *Distributed consensus algorithms in sensor networks: quantized data and random link failures*, IEEE Transactions on Signal Processing **58** (2010), no. 3, 1383–1400. (Cited in pages 72 and 73.)

[KMT98] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, *Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability*, The Journal of the Operational Research Society **49** (1998), no. 3, 237–252. (Cited in pages 3 and 147.)

[KTB13] Ali Khanafer, B. Touri, and Tamer Başar, *Robust distributed averaging on networks with adversarial intervention*, Proceedings of 52nd IEEE Conference on Decision and Control (IEEE CDC), December 2013. (Cited in page 58.)

[KVV00] R. Kannan, S. Vempala, and A. Veta, *On clusterings-good, bad and spectral*, Proceedings of the 41st Annual Symposium on Foundations of Computer Science (Washington, DC, USA), FOCS, 2000, p. 367. (Cited in page 124.)

[LFXZ11] Tao Li, Minyue Fu, Lihua Xie, and Ji-Feng Zhang, *Distributed consensus with limited communication data rate*, IEEE Transactions on Automatic Control **56** (2011), no. 2, 279–292. (Cited in page 72.)

[LM06] Amy N. Langville and Carl D. Meyer, *Google's pagerank and beyond: The science of search engine rankings*, Princeton University Press, Princeton, NJ, USA, 2006. (Cited in pages 3 and 147.)

[LM12] J. Lavaei and R. M. Murray, *Quantized consensus by means of gossip algorithm*, IEEE Transactions on Automatic Control **57** (2012), no. 1, 19–32. (Cited in pages 72 and 73.)

[LNR09] Yao Liu, Peng Ning, and Michael K. Reiter, *False data injection attacks against state estimation in electric power grids*, Proceedings of the 16th ACM Conference on Computer and Communications Security (New York, NY, USA), CCS '09, ACM, 2009, pp. 21–32. (Cited in page 58.)

[LO81] H.J. Landau and A.M. Odlyzko, *Bounds for eigenvalues of certain stochastic matrices*, Linear Algebra and its App. **38** (1981), no. 0, 5 – 15. (Cited in page 126.)

[LO11] Ilan Lobel and Asuman E. Ozdaglar, *Distributed Subgradient Methods for Convex Optimization Over Random Networks*, IEEE Transactions on Automatic Control **56** (2011), no. 6, 1291–1306. (Cited in page 32.)

[LSch] Jia Liu and H.D. Sherali, *A distributed newton's method for joint multi-hop routing and flow control: Theory and algorithm*, Proceedings of IEEE INFOCOM 2012, March, pp. 2489–2497. (Cited in page 25.)

[LSB+03] David Lusseau, Karsten Schneider, Oliver J. Boisseau, Patti Haase, Elisabeth Slooten, and Steve M. Dawson, *The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations*, Behavioral Ecology and Sociobiology **54** (2003), no. 4, 396–405. (Cited in pages 5, 38 and 42.)

[LVS12] Frank L. Lewis, Draguna Vrabie, and Vassilis L. Syrmos, *Optimal control. 3rd ed.*, Hoboken, NJ: John Wiley Sons. , 2012. (Cited in page 61.)

[LX11] T. Li and L. Xie, *Distributed consensus over digital networks with limited bandwidth and time-varying topologies*, Automatica **47** (2011), no. 9, 2006–2015. (Cited in page 72.)

[Mey00] Carl D. Meyer, *Matrix analysis and applied linear algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. (Cited in pages 8 and 20.)

[Nes04] Yurii Nesterov, *Introductory lectures on convex optimization : a basic course*, Applied optimization, Kluwer Academic Publ., Boston, Dordrecht, London, 2004. (Cited in page 38.)

[NFZE07] G. Nair, F. Fagnani, S. Zampieri, and R. Evans, *Feedback control under data rate constraints: an overview*, Proceedings of The IEEE **95** (2007), 108–137. (Cited in page 74.)

[NG04] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E **69** (2004), 026113. (Cited in pages 124 and 132.)

[NO09] A. Nedić and A. Ozdaglar, *Distributed Subgradient Methods for Multi-Agent Optimization*, IEEE Transactions on Automatic Control **54** (2009), no. 1, 48–61. (Cited in pages 3, 32 and 147.)

[NOOT09] Angelia Nedic, Alex Olshevsky, Asuman Ozdaglar, and John N. Tsitsiklis, *On distributed averaging algorithms and quantization effects*, IEEE Transactions on Automatic Control **54** (2009), no. 11, 2506–2517. (Cited in pages 72 and 74.)

[ORG12] PederA. Olsen, StevenJ. Rennie, and Vaibhava Goel, *Efficient automatic differentiation of matrix functions*, Recent Advances in Algorithmic Differentiation, Lecture Notes in Computational Science and Engineering,

vol. 87, Springer Berlin Heidelberg, 2012, pp. 71–81 (English). (Cited in page 54.)

[OSFM07] R. Olfati-Saber, J.A. Fax, and R.M. Murray, *Consensus and cooperation in networked multi-agent systems*, Proceedings of the IEEE **95** (2007), no. 1, 215–233. (Cited in pages 4 and 148.)

[OSM04] R. Olfati-Saber and R.M. Murray, *Consensus problems in networks of agents with switching topology and time-delays*, IEEE Transactions on Automatic Control **49** (2004), no. 9, 1520–1533. (Cited in page 141.)

[Pen03] Mathew Penrose, *Random Geometric Graphs*, Oxford University Press, USA, July 2003. (Cited in page 5.)

[Pol87] B.T. Polyak, *Introduction to optimization*, Optimization Software, New York, 1987. (Cited in pages 32 and 36.)

[SA04] J. Shetty and J. Adibi, *The Enron email dataset database schema and brief statistical report*, Information sciences institute technical report, University of Southern California, 2004. (Cited in pages 5, 38, 42 and 46.)

[Sch64] L. Schuchman, *Dither signals and their effect on quantization noise*, IEEE Transactions on Communication Technology **12** (1964), no. 4, 162–165. (Cited in pages 72 and 73.)

[Sch07] Satu Elisa Schaeffer, *Survey: Graph clustering*, Comput. Sci. Rev. **1** (2007), no. 1, 27–64. (Cited in page 124.)

[SECN13] Lorenzo Severini, Mahmoud El Chamie, and Giovanni Neglia, *Topology versus link strengsth for information dissemination in networks*, ALGOTEL 2013 (Pornic, Loire-Atlantique, France, May 28-31), May 2013, p. 4. (Cited in pages 11 and 149.)

[Sen06] E. Seneta, *Non-negative matrices and markov chains*, Springer Series in Statistics, Springer, 2006. (Cited in page 49.)

[SH07] S. Sundaram and C.N. Hadjicostis, *Finite-time distributed consensus in graphs with time-invariant topologies*, American Control Conference (ACC '07), July 2007, pp. 711 –716. (Cited in page 104.)

[Sho85] N.Z. Shor, *Minimization methods for non-differentiable functions*, Springer Series in Computational Mathematics, Springer, Berlin, 1985. (Cited in page 63.)

[SM12] V. Schwarz and G. Matz, *Mean-square optimal weight design for average consensus*, 2012 IEEE 13th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2012, pp. 374–378. (Cited in page 57.)

[SRJ05] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakola, *Maximum-margin matrix factorization*, Advances in Neural Information Processing Systems 17, MIT Press, 2005, pp. 1329–1336. (Cited in page 52.)

[SS94] R. Subramanian and I. D. Scherson, *An analysis of diffusive load-balancing*, Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures, 1994, pp. 220–225. (Cited in page 73.)

[TKPF13] D. Thanou, E. Kokiopoulou, Y. Pu, and P. Frossard, *Distributed average consensus with quantization refinement*, IEEE Transactions on Signal Processing **61** (2013), no. 1, 194–205. (Cited in page 72.)

[TSJ08] A Tahbaz-Salehi and A Jadbabaie, *A necessary and sufficient condition for consensus over random networks*, IEEE Transactions on Automatic Control **53** (2008), no. 3, 791–795. (Cited in page 141.)

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to data mining, (first edition)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. (Cited in page 124.)

[VD08] Stijn Van Dongen, *Graph clustering via a discrete uncoupling process*, SIAM J. Matrix Anal. Appl. **30** (2008), no. 1, 121–141. (Cited in page 124.)

[VD14] O. Vukovic and G. Dan, *Security of fully distributed power system state estimation: Detection and mitigation of data integrity attacks*, IEEE Journal on Selected Areas in Communications **32** (2014), no. 7, 1500–1508. (Cited in page 58.)

[Wes00] Douglas B. West, *Introduction to Graph Theory (2nd Edition)*, Prentice Hall, August 2000. (Cited in page 33.)

[WOEJ12] Ermin Wei, A. Ozdaglar, A. Eryilmaz, and A. Jadbabaie, *A distributed newton method for dynamic network utility maximization with delivery contracts*, 46th Annual Conference on Information Sciences and Systems (CISS), 2012, pp. 1–6. (Cited in page 25.)

[WOJ13] Ermin Wei, A. Ozdaglar, and A. Jadbabaie, *A distributed newton method for network utility maximization, I: algorithm*, IEEE Transactions on Automatic Control **58** (2013), no. 9, 2162–2175. (Cited in pages 3 and 147.)

[XB04] Lin Xiao and Stephen Boyd, *Fast linear iterations for distributed averaging*, Systems and Control Letters **53** (2004), 65–78. (Cited in pages 7, 9, 10, 17, 18, 19, 22, 53, 57, 59 and 69.)

[XBK07] Lin Xiao, Stephen Boyd, and Seung-Jean Kim, *Distributed average consensus with least-mean-square deviation*, J. Parallel Distrib. Comput. **67** (2007), no. 1, 33–46. (Cited in pages 19 and 111.)

[YFG+08] Peng Yang, R.A. Freeman, G.J. Gordon, K.M. Lynch, S.S. Srinivasa, and R. Sukthankar, *Decentralized estimation and control of graph connectivity in mobile sensor networks*, American Control Conference (ACC 2008), June 2008, pp. 2678 –2683. (Cited in page 19.)

[YS07] Vikas Yadav and Murti V. Salapaka, *Distributed protocol for determining when averaging consensus is reached*, Forty-Fifth Annual Allerton Conference Allerton House, UIUC, Illinois, USA, September 2007. (Cited in page 104.)

[Zac77] W.W. Zachary, *An information flow model for conflict and fission in small groups*, J. of Anthropological Research **33** (1977), 452–473. (Cited in page 131.)

[ZM11] M. Zhu and S. Martínez, *On the convergence time of asynchronous distributed quantized averaging algorithms*, IEEE Transactions on Automatic Control **56** (2011), no. 2, 386–390. (Cited in page 73.)

[ZZ13] Q. Zhang and J. F. Zhang, *Quantized data-based distributed consensus under directed time-varying communication topology*, SIAM Journal on Control and Optimization **51** (2013), no. 1, 332–352. (Cited in page 72.)