Distributed Consensus in Networks

Arta Babaei

Supervised by Dr. Moez Draief

Submitted in part fulfilment of the requirements for the degree of Doctor of Philosophy in Department of Electrical and Electronic Engineering of Imperial College London and the Diploma of Imperial College London
Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.
Abstract

Distributed algorithms have gained a lot of attention during recent years. Their application in industry, particularly in wireless sensor networks has motivated researchers to try to design them in order to be less resource-consuming (e.g. memory and power), faster, and more reliable. There have been numerous distributed algorithms for different types of problems in the context of distributed algorithms. We are interested in a fundamental coordination problem namely the majority consensus problem. In the majority consensus problem nodes try to find the opinion of the majority in a network of interest. As our first contribution and motivated by the distributed binary consensus problem in [1] we propose a distributed algorithm for multivalued consensus in complete graphs. As our second contribution we propose an algorithm for the optimization of the binary interval consensus algorithm pioneered by Benézit et al in [2]. Finally we use binary interval consensus algorithm to design a framework for error-free consensus in dynamic networks using which nodes can leave or join the network during or after the consensus process.
To my parents, Shadan and Hassan
Acknowledgement

Firstly, I wish to express my deepest gratitude to my supervisor, Dr. Moez Draief, whom I have been fortunate to have known. During the period of my studies as a MSc and later on a PhD student he has always been patient and given me continuous support. For that and the opportunity that he has given me I will be forever indebted to him.

I want to thank my examiners Dr. Tim Evans, and Professor Colin Cooper for their invaluable feedback on my thesis.

I also wish to thank my colleagues, Niki, Sing, and Julia who have been a great support all along the way. During my PhD studies I have had the pleasure of conducting extracurricular activities which helped me to form a new perspective with regards to research. This would not have been possible without Pedro, Tola, Mohammadreza and Kasra.

I would like to thank my parents for their unstoppable and immeasurable belief in me and their support throughout my studies. I hope that they continue believing in me.

Finally, I would like to thank my best friend and partner Parisa for all her love and unconditional support. It is due to her that I am here today.
Contents

Copyright Declaration

Abstract

Acknowledgement

1 Introduction

1.1 Statement of originality

2 Notations, definitions, and preliminaries

2.1 Notations

2.1.1 A note on the numbering of choices in different types of consensus

2.2 Graphs

2.2.1 Erdős-Rényi

2.2.2 Complete

2.2.3 Regular

2.2.4 Path

2.2.5 Grid
2.2.6 Power-law .................................................. 25

2.3 Time models preliminaries ..................................... 27
  2.3.1 Synchronous model ...................................... 28
  2.3.2 Asynchronous model ..................................... 28
  2.3.3 Absolute time and clock ticks .......................... 28

3 Related literature ................................................. 30
  3.1 Binary Consensus using the voter model ................. 31
    3.1.1 Algorithm .............................................. 32
    3.1.2 Probability of error .................................... 33
  3.2 Binary consensus using three states .................... 34
    3.2.1 Dynamics ................................................. 35
    3.2.2 Probability of error .................................... 39
    3.2.3 Convergence time ....................................... 41
  3.3 Binary Consensus using four states (binary interval consensus) 42
    3.3.1 Dynamics ................................................. 43
    3.3.2 Convergence time ....................................... 50
  3.4 Fastest mixing Markov chain problem .................... 52
    3.4.1 Markov Chain on an undirected graph \( G(V, E) \) .... 52
    3.4.2 Fastest mixing problem ................................. 53
    3.4.3 Convexity of \( \zeta(P) \) ................................ 54
    3.4.4 SDP formulation ......................................... 56
    3.4.5 Sub-gradient of \( \zeta(P) \) ............................... 57
    3.4.6 Optimization ............................................ 58
  3.5 Distributed averaging and optimization of convergence time 62
7 Conclusion and future work 118
   7.1 Multivalued consensus 118
   7.2 Optimization of binary interval consensus 119
   7.3 Binary interval consensus in dynamic networks 120
   7.4 Future work 120

Bibliography 123

Appendices 131

A First step analysis 132
   A.1 Gambler’s ruin 132

B Density dependent jump Markov processes 135

C Previous attempt on finding the lower bound of $\delta(G, \alpha)$ 138
   C.1 Problems with the analysis 142
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>An Erdős-Rényi graph of size 20 where $p = 0.5$</td>
<td>22</td>
</tr>
<tr>
<td>2.2</td>
<td>A regular graph of size 20 and degree 4</td>
<td>23</td>
</tr>
<tr>
<td>2.3</td>
<td>A path graph of size 10</td>
<td>24</td>
</tr>
<tr>
<td>2.4</td>
<td>A grid of size 16</td>
<td>25</td>
</tr>
<tr>
<td>2.5</td>
<td>A power-law graph of size 20</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Sample of a voter model process</td>
<td>33</td>
</tr>
<tr>
<td>3.2</td>
<td>Binary Consensus using three states. Nodes start with colours white and black. Undecided nodes are denoted by grey colour</td>
<td>37</td>
</tr>
<tr>
<td>3.3</td>
<td>Simple graph with self-loops</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Binary interval consensus. Nodes start with colours white and black. Colours light grey and dark grey denote states $0.5^-, 0.5^+$ respectively</td>
<td>49</td>
</tr>
</tbody>
</table>
4.1 Simulation results for $k = 3$ for Complete graphs. In the error plot (top) $\alpha_1 = 0.4, \alpha_2 = 0.35, \alpha_3 = 0.25$ and in the Convergence time plot (bottom) $\alpha_1 = 0.7, \alpha_2 = 0.2, \alpha_3 = 0.1$. The bars indicate %90 confidence estimates. Upper bound is given by (4.7). 

4.2 Simulation results for $k = 3$ for Erdős-Rényi graphs. In the error plot (top) $\alpha_1 = 0.4, \alpha_2 = 0.35, \alpha_3 = 0.25$ and in the Convergence time plot (bottom) $\alpha_1 = 0.7, \alpha_2 = 0.2, \alpha_3 = 0.1$. The bars indicate %90 confidence estimates. Upper bound is given by (4.7). 

4.3 Simulation results for $k = 3$ for regular graphs. In the error plot (top) $\alpha_1 = 0.4, \alpha_2 = 0.35, \alpha_3 = 0.25$ and in the Convergence time plot (bottom) $\alpha_1 = 0.7, \alpha_2 = 0.2, \alpha_3 = 0.1$. The bars indicate %90 confidence estimates. Upper bound is given by (4.7). 

5.1 Simulation of $\delta(G, \alpha)$ and its bound (5.1) for Erdős-Rényi and regular graphs. 

5.2 Simulation of $\delta(G, \alpha)$ and its bound (5.1) for Power-law and Grid graphs. 

5.3 Simulation of the convergence time and its upper bound (5.2) for Erdős-Rényi and regular graphs. 

5.4 Simulation of the convergence time and its upper bound (5.2) for Power-law and Grid graphs.
5.5 Simulation results for optimized and non-optimized versions of binary interval consensus for a grid of size 144. The bars indicate %90 confidence estimates.

5.6 Simulation results for optimized and non-optimized versions of binary interval consensus. (Top) Expected time for an Erdős-Rényi graph of size 400. (Bottom) Expected time for a regular graph of size 324. The bars indicate (at least %90) confidence estimates.

5.7 Simulation results for optimized and non-optimized versions of binary interval consensus. (Top) Expected time for a Power law graph of size 225 with $h = 3$. (Bottom) Expected time for a Power law graph of size 225 with $h = 3.5$. The bars indicate %80 confidence estimates.

5.8 Two simple star graphs connected by one edge. Note that self-loops have not been shown in the figure.

5.9 Simulation results for optimized and non-optimized versions of binary interval consensus. (Top) Probability matrix before the optimization and (Bottom) probability matrix after optimization.

5.10 Simulation results for optimized and non-optimized versions of binary interval consensus process in a graph of two connected stars.

5.11 Optimization of $\lambda_2(Q)$ for an Erdős-Rényi graph.
Chapter 1

Introduction

In numerous network settings, no central controller is available to coordinate the computations between the nodes. In fact, gathering, managing and processing the data held by individual nodes might consume a considerable amount of resources (e.g. time and energy), which is not always feasible. In many such networks, the aim of the nodes is to calculate a function of the initial (or current) values of the nodes in a distributed fashion (i.e. nodes carrying out some internal computations and contacting each other in an iterative manner). Furthermore, using distributed algorithms allows a system to become more resilient as the failure of individual nodes will not effect the computations as long as the graph stays connected (although the result of the distributed process might change).

There has been recently an ever growing interest in devising efficient distributed computation procedures in networks. The performance of such algorithms is based on the analysis of their accuracy, reliability, and speed of
convergence. Since agents are assumed to be unaware of the underlying network and their knowledge of the network is restricted to their neighbours, guaranteeing the efficiency of such distributed computations is a rather challenging task. Addressing this problem is paramount in areas as diverse as distributed databases, social networks, and wireless sensor networks.

Today, major Telecom companies such as Cisco and Ericsson are using distributed algorithms to let industry managers gain better insight into complex processes by securely networking sensor data. Some of the main challenges include improving resource consumption, improving efficiency of operations, optimizing maintenance, and asset surveillance [3].

Formally with the distributed algorithms the problem statement is as follows: Given a graph $G(V, E)$ (where the set of vertices is denoted by $V$ and the set of edges is denoted by $E$) we are interested in computing a function for each node (or agent) ($F_i$) which depends on ($X_1, X_2, \ldots, X_n$) where $X_i$ is a variable corresponding to node $i$. This function can be an average amount of a certain quantity, number of nodes, or the majority’s opinion.

Our focus in this thesis is on a fundamental coordination problem, namely the distributed consensus problem. With the distributed consensus problem, nodes use local interactions with their neighbours to reach an agreement on a choice from a set of $k$ available choices. This reduces to the binary consensus problem if the set of choices is limited to two ($k = 2$). A typical case where the distributed consensus algorithms can be used is in cooperative decision making. Such applications can arise in surveillance and security, environmental monitoring [4], and distributed voting [5].
There have been few distributed algorithms proposed to solve the distributed consensus problem that we discuss below.

Probably the most famous distributed consensus algorithm is the voter model. The algorithm has been investigated extensively in [6]. In the asynchronous voter model initially nodes in the network hold one of the two states (0 or 1) and then at each time step one of the nodes contacts one of its neighbours and simply adopts its opinion. This process continues until all the nodes are in the same state. It is proved in [6] that the voter model always converges. However, the probability of error, i.e. the probability of converging to the incorrect consensus (e.g. when all the nodes hold 0 at the end of the process while the initial majority was 1), depends on the degree of the nodes which are initially in the minority. For instance for the complete graph, this results in the probability of error equal to the portion of nodes in the minority state.

To address this problem in [1] an additional undecided state e was introduced. Nodes adopt this new state e if they happen to be in disagreement with their polled neighbour. It is proved that this additional state reduces the probability of error and convergence time of the consensus algorithm significantly. More precisely, when the graph is complete, adding this additional state results in an exponential decay of the probability of error with the size of the network (i.e. N) at a rate that depends on the portion of the majority.

In the literature related to the distributed consensus problem the focus has usually been on binary consensus (e.g. [7], [11]) and there have been few results other than the voter model itself regarding the distributed consensus on more than two choices. One exception is [8] in which Bénézit et al have introduced
a majority consensus algorithm for the case of $k \geq 2$ choices for general connected graphs. While their algorithm guarantees the correct consensus it needs 4, 15, and 100 memory and communication states for the cases of $k = 2, 3, \text{ and } 4$ respectively. This means that the nodes should be able to store and communicate 100 values for a consensus on 4 choices.

As our first contribution in Chapter 4 and based on the distributed binary consensus algorithm in [1] we introduce a distributed multivalued consensus algorithm which has a superior performance relative to the voter model (in terms of both probability of error and convergence time) while using only one more state. We use the same setup for consensus on $k$ choices, which means adding only one state (in addition to the voter model) for both communication and memory states. We prove that using this additional state, the convergence time becomes logarithmic in $N$ (i.e. the number of the nodes of the graph) for large $N$ in the case of complete graphs. We also show that the upper bound on the error probability decreases exponentially with a rate that depends on both $N$ and the fraction of the two choices with highest number of votes. We then confirm these findings with simulations. Chapter 4 is based on the joint work with Moez Draief published in the Computer Journal [9] (with the first version published in [10]).

Although adding an undecided state improves the error probability and convergence time significantly such asymptotic correctness is only proved in the case of complete graph. In particular, for path graphs, the addition of the undecided state yields the same performance as the voter model with respect to the probability of error as well as a slower convergence.
In [7], a special case of quantized consensus (first introduced in [2]) is studied. This algorithm is proved to converge to the right result with probability one. However, the time of convergence is $O\left(\frac{1}{\delta(G,\alpha)} \log N\right)$ where $\delta$ depends on both the graph structure and $\alpha$ the fraction of the initial majority ($0.5 < \alpha < 1$). For instance, it is proved that in the case of complete graphs $\delta = 2\alpha - 1$.

As our second contribution in Chapter 5 we show that if the probability of the nodes contacting each other is defined by a doubly stochastic matrix $P$, the optimization of binary interval consensus can be performed by reducing the second largest eigenvalue of the rate matrix $Q$ which governs the binary interval consensus. We use eigenvalue optimization techniques to reduce the second largest eigenvalue of $Q$ and hence reduce the convergence time of the binary interval consensus. Chapter 5 is based on the joint work with Moez Draief published in [11].

In a real network nodes might have to leave or join the network. In such a network, reaching the correct consensus can become problematic. By correct consensus we mean each node has to decide which one of the states was initially held by the majority of the nodes present in the network without being affected by the decision of the nodes which have left the network. This decision might have to change depending on the state of the nodes which leave or join the network frequently.

Using the conventional distributed algorithms, all nodes have to restart the consensus algorithm after each time a node joins or leaves the network. This is because normally after running the distributed algorithm, all the nodes change their initial value to the final value. For instance, using the algorithms
in [6] and [1], after reaching the consensus all the nodes are either in state 1 or 0. In a network using these algorithms for reaching consensus, each node has to store its initial state and all the nodes have to restart the process to reach the correct consensus every time a node joins or leaves the network.

Needless to say, restarting the algorithm every time a node joins or leaves the network will be time and energy consuming. Another problem with these methods is that the nodes will not know when to restart their algorithm as these are decentralized settings and each node only has the local knowledge of its neighbours. One solution might be to run the algorithms after a certain period of time, however, this is a weak solution as there is no specific decentralized measure to find this period. Moreover, consensus algorithms in [6] and [1] are not error-free.

It is worth mentioning the recent work in [12] which investigates the distributed averaging in dynamic networks as we can consider binary consensus as a special case of averaging problem (where the majority is 1 when the average is more than 0.5). In [12] the bound on the difference between the actual average and the estimates (i.e. the error) has been found. This has been done for different types of changes in the values (namely as multiplicative changes, and Additive changes) in different types of communication (Probabilistic and Deterministic). Also, changes in topology in terms of adding or removing edges and joining or leaving nodes have been considered. However, there has not been any proposition to reduce the error of the estimates.

As our last contribution and using binary interval consensus algorithm, in Chapter 6 we suggest a framework for the majority consensus in dynamic
networks. More specifically, we aim for finding a consensus algorithm in which nodes can join or leave the network at any time (even during the process of reaching the consensus) without causing any errors in the final result. Obviously the final majority will be amongst the nodes which are present in the network. Chapter 6 is based on the joint work with Moez Draief which was accepted as part of the IASTED conference in 2012 [13].

1.1 Statement of originality

This thesis is submitted for the degree of Doctor of Philosophy in the Department of Electrical and Electronic Engineering at Imperial College London. The research work reported herein was carried out, unless otherwise stated, by the author in the Department of Electrical and Electronic Engineering at Imperial College London between Nov 2009 and March 2013. No part of this thesis has been submitted in support of an application for a degree or qualification of this or any other university or educational establishment.
Chapter 2

Notations, definitions, and preliminaries

In this chapter we give the notations and definitions which we will use throughout the thesis. The main focus of this thesis is on the distributed algorithms. Whether we are discussing the averaging problem in wireless sensor networks (Chapter 3) or a distributed majority consensus algorithm we are always dealing with a scenario in which nodes try to calculate a particular target value using local interactions with the neighbours.

2.1 Notations

We consider a graph $G$ which is defined by the two sets $(V, E)$ where $V$ is the set of vertices and $E$ denotes the set of edges. If $(i, j) \in E$ nodes $i, j$ are neighbours (adjacent). The degree of node $i$ is then denoted by $d_i$ and is the
2.1 Notations

number of node $i$’s neighbours.

Furthermore, in the distributed algorithms that we discuss in the following chapters nodes contact their neighbours based on certain probabilities. Concretely, the probability of node $i$ contacting $j$ is denoted by $P_{ij}$. Clearly, $P_{ij} = 0$ when $(i, j) \notin E$.

While the graphs we consider are always assumed to be connected the edges might be directed or undirected based on the framework of the distributed algorithm. This can be recognised from the probability matrix $P$ and whether $P_{ij}$ is equal to $P_{ji}$ (undirected) or not (directed).

If $V = \{1, \ldots, N\}$, the adjacency matrix of the graph, $A = [A_{ij}]$ is defined by a $N \times N$ matrix in which $A_{ij} = 1$ if $(i, j) \in E$ and zero otherwise.

2.1.1 A note on the numbering of choices in different types of consensus

Throughout the thesis there will be a slight difference in the naming of the choices of different types of consensus algorithms (e.g. 1 and 2 vs. 0 and 1). This is solely due to the simplification of notations in proofs and consistency. Therefore, in Chapter 4 where we consider the consensus on $k$ choices namely $1, \ldots, k$ it makes sense to start the number of our choices from 1. We also use this way of numbering the choices in Section 3.2 (i.e. choice 1 and 2) as it discusses the binary consensus algorithm which will be used later in Chapter 4.
In the rest of the thesis where we are only focusing on two choices we choose the notation 0, 1 so that we can relate the binary consensus to an averaging problem where the average being greater or less than 0.5 will define the majority. Using this notation will be specifically useful in Chapter 6 where the sum of the values of the nodes of the graph becomes very important in finding the correct consensus result.

2.2 Graphs

Throughout the thesis we consider different types of graphs namely complete, Erdös-Rényi, regular, grid, Power-law and path for the sake of simulations and analysing the performance of different distributed algorithms. It is therefore useful to give a brief overview of their characteristics.

2.2.1 Erdös-Rényi

Erdös-Rényi graphs were first introduced by Paul Erdös and Alfred Rényi. The most common model of Erdös-Rényi networks is defined using two parameters \( n \) and \( p \) in \( G(n, p) \) where the former is the number of nodes and the latter is the probability of existing an edge between any pair of nodes in the graph. Considering the number of neighbours of each vertex as the degree \( (d) \) of that particular vertex, the average degree \( \bar{d} \) in an Erdös-Rényi graph is \[ \bar{d} = \frac{N(N-1)p}{N} = (N-1)p \]
2.2 Graphs

2.2.2 Complete

In the case of complete graphs each node is connected to every other node and therefore there is an edge between any pair of nodes, equivalently,

\[ A(i, j) = 1, \forall i, j \in V \]

Accordingly, the number of edges in a complete graph is \( \binom{N}{2} \).
2.2.3 Regular

A \textit{d-regular} graph is a graph which comprises of nodes of the same degree \( d \) (e.g. \cite{15}). A regular graph can be generated by the configuration model in \cite{16}. Our implementation of the configuration model is the modified erased configuration model in \cite{17} which gives a good approximation of a random regular graph.

![Regular Graph](image)

Figure 2.2: A regular graph of size 20 and degree 4
2.2 Graphs

2.2.4 Path

For the case of path graphs, each of the vertices is part of an open chain. Therefore, nodes can be numbered in a way that

\[ A_{ij} = \begin{cases} 
A(i, i + 1) = A(i, i - 1) = 1, & \forall 1 < i < N \\
0, & \text{elsewhere}
\end{cases} \]

We use the path graph in Chapter 4 as a special case where using more resources worsens the performance of a distributed algorithm.

![Figure 2.3: A path graph of size 10](image)

2.2.5 Grid

In this thesis a square grid (or simply grid) is a lattice graph on 2D plane or simply a Cartesian product of two path graphs. Lattice graphs have been used extensively within different contexts of research (e.g. [18], [19]). In Chapter 5 we use grid as a graph with a given degree distribution to analyse
the performance of our proposed optimization method for a specific type of consensus process.

![Graph of size 16](image)

Figure 2.4: A grid of size 16

### 2.2.6 Power-law

To further analyse the performance of our proposed optimization method in Chapter 5, we also use a different type of graph in which the portion of nodes with degree $d_j$ is proportional to $d_j^{-h}$ ($h > 2$). This type of graphs are known as Power-law. The degree distribution of the Power-law graphs is very different from those of Erdős-Rényi and regular and can give a better understanding of the performance of our optimization method for graphs which are not homogeneous. The concept of Scale free networks or Power-law graphs have first been introduced in the work of Derek de Solla Price in 1965 [20] and later in 1976 [21] although he did not use the term ”scale-free”. Power-law graphs have been used to model wide range of practical networks including internet, cities, and people’s wealth (e.g. [22]).
2.2 Graphs

To generate power-law graphs we use the well-known preferential attachment algorithm ([23], [24]). Starting with an initial graph, using preferential attachment the graph expands over time by adding nodes where a new node $i$ attaches to the node $j$ of the previous graph based on its degree (with probability $\frac{d_i}{\sum_k d_k}$).

In our simulations we use a modified version of the preferential attachment method that can give a wider range of Power-law graphs. Concretely a new parameter $\nu$ is defined (e.g. [25]); With probability $1-\nu$ the new node $i$ attaches to node $j$ preferentially (with probability $\frac{d_j}{\sum_k d_k}$) and with probability $\nu$ it chooses one of the nodes uniformly at random. The exponent of the graph (i.e. $h$) will be given by the following [25],

$$h = 3 - \frac{\nu}{1 - \nu}$$

We will use this relationship to generate Power-law graphs with different powers in Chapter 5.
Throughout this thesis we go over different problems in the context of distributed algorithms. However, our analysis is always under the assumption that nodes contact each other under a \textit{gossip constraint}. By this we mean that in any of the distributed algorithms, we consider the case where each node can only contact one neighbour at each time step. In the literature of gossip constrained algorithms the following time models are usually considered.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{power-law-graph.png}
\caption{A power-law graph of size 20}
\end{figure}
2.3.1 Synchronous model

In the synchronous model a global clock exists using which at every tick each node contacts one of its neighbours based on a probability distribution and then performs a certain procedure (polling, averaging, etc.). This means that all the nodes contact their neighbours at the same time.

2.3.2 Asynchronous model

With asynchronous model, each node has a clock which ticks with a certain rate. The time horizon is then divided according to these ticks and at any one time a node contacts one of the neighbours based on a specific probability distribution. Here, in contrast with the synchronous model, at each time slot only one node contacts its neighbour and performs a local computation.

2.3.3 Absolute time and clock ticks

Concretely, consider the case where the tick-rate of the clock of each node is Poisson with $\lambda = 1$. Note that for a Poisson process of rate $\lambda$ and for any time $t = T_{k+1} - T_k > 0$, the probability of having $n$ occurrence in time interval $(T_k, T_{k+1}]$ is given by the following (e.g. [26]),

$$P_n = \frac{e^{-\lambda t} (\lambda t)^n}{n!}.$$  \hfill (2.1)

In such a setting (e.g. consensus in [1] and averaging in [27]) this will be
equivalent of a global clock of Poisson rate $N$ at times $T_k$. The time intervals 
$\{T_{k+1} - T_k\}$ are exponentials with rate $N$ \cite{27}. This means that on average 
at each time unit there are $N$ ticks and consequently $N$ contacts between 
the nodes. The following lemma is then proved in \cite{27},

**Lemma 1.** *The relationship between the ticks and the absolute time is given by,*

$$\mathbb{E}[T_k] = \frac{k}{N}, \forall k \geq 1$$

*Furthermore, the following relationship exists,*

$$\mathbb{P}(|T_k - k/N| \geq \delta k/N) \leq 2e^{-\delta^2 k/2}, \forall \delta > 0$$
Chapter 3

Related literature

In this chapter we present the related work with regards to our research. We first give an overview of three different distributed consensus algorithms which comprise the foundation of our research. We start by discussing the voter model and the results regarding its probability of error in Section 3.1. Though light-weight and simple, the voter model is the basis of many other distributed algorithms in the literature and is a perfect example of a distributed solution to the consensus problem.

Next, we give an overview of a distributed algorithm for binary consensus problem (in complete graphs) proposed by Vojnovic et al in [1]. This algorithm, which uses additional memory and communication state, gives a better performance relative to the voter model and is the starting point of our solution for the multivalued consensus problem in Chapter 4.

We then give an overview of the binary interval consensus algorithm in [2] which uses two additional states (relative to the voter model) for binary con-
3.1 Binary Consensus using the voter model

Binary consensus problem can be viewed as a simpler case of the averaging problem in which nodes are only interested to know if the average of all the
values is higher or lower than 0.5. D. Peleg and Y. Hassin in [6], investigate a local probabilistic polling process (used for reaching consensus) and its properties. The algorithm is very simple and easy to implement, however, it is not accurate and has a relatively high probability of error.

### 3.1.1 Algorithm

Initially each node holds one of the states of 1 and 0 (or coloured in white or black), and the goal is for all the nodes to eventually have an agreement on one of the two states. In the model considered, there is a global clock that ticks at discrete points of time for all the nodes and at these time steps each node will perform a local polling process in which it chooses one of its neighbours with probability $P_{ij}$ (weight of the edge between $i$ and $j$) and changes its opinion to the opinion of that neighbour. Figure 3.1.1 shows the evolution of the binary consensus using voter model in a simple graph of five nodes starting with two colours (i.e. white and black).
3.1 Binary Consensus using the voter model

3.1.2 Probability of error

For such an algorithm it is then proved that:

- With probability equal to 1, all the nodes will have the same state after a finite time period (finite number of steps).

- If the probability matrix $P$ is ergodic with stationary distribution $\pi$ ($\pi P = \pi$) then the nodes will be eventually in state 1 with probability proportional to $\sum_{i \in W} \pi_i$ where $W$ is the set of nodes initially in state 1. As a special case, when nodes choose their neighbour with uniform

\footnote{In this setting the number of nodes is finite.}
probability $\pi_i$ is equal to $\frac{d_i}{2m}$ (i.e. uniform random walk), where $m$ is the number of edges and $d_i$ denotes the degree of node $i$. In such a case, the probability of all nodes ending in state 1 is $\sum_{i \in W} \frac{d_i}{2m}$.

This will then mean that depending on the proportion of ones and zeros and the structure of the graph, the error margin might become significant and therefore using the voter model is not the best way when accuracy is needed.

It should be noted that this is a synchronous model and dynamics are different than those of an asynchronous model where at each time step only one node contacts one of its neighbours. However, as pointed out in [6] the asynchronous model has been investigated in [32] and the probability to end up in the all one state is the same.

### 3.2 Binary consensus using three states

In [1], a binary consensus algorithm is introduced which uses three states for communication and memory. This is one state more than the voter model with two choices. It is proved that by using this additional state the probability of error decreases exponentially with the number of nodes and the convergence time becomes logarithmic in $N$ for the case of complete graphs.

As before, with the binary consensus problem each node initially decides on one of the two choices and the aim for all the nodes is to reach an agreement on the choice which was initially held by the majority. Now consider an undirected graph $G(V, E)$. We first go over the dynamics of the algorithm...
and then a give a brief overview of the results found in [1] with respect to its convergence time and the probability of error.

### 3.2.1 Dynamics

Consider an asynchronous setup which is usually used in the context of distributed algorithms (e.g. [27]). In such a setup, each node has a clock which ticks at the instances of a Poisson process with rate 1. At each time step, when a node’s (node $z$) clock ticks, it picks one neighbour $w$ with probability $P_{zw}$. As a result, if the graph is complete, and the probability distribution of the interactions is uniform, the neighbour is chosen from the set $V$ with probability $P_{zw} = \frac{1}{N-1}$. Node $z$ will then change its state according to the message received from node $w$.

The decision process of each node is as follows; let nodes initially choose one of the two states 1 and 2 (note the difference in the naming of the choices, refer to Subsection 2.1.1). Also, consider a state named $e$, an additional state (compared with the voter model) which shows that the node is undecided on which of the two states 1 and 2 was initially held by the majority.

If a node in state $i$ ($i \in \{1,2\}$) contacts a node in state $i$ or $e$, it does not change its state and if it contacts a node in state $j$ ($j \in \{1,2\}, i \neq j$), it updates its value to $e$. Also, if a node in the undecided state $e$ contacts a node in state $i$, it changes its state to $i$.

In other words, nodes that have already decided about the initial majority become undecided following a contact with a neighbour with a different
opinion. Furthermore, any undecided node simply accepts the opinion of its neighbour (similar to the voter model). Figure 3.2 shows a simple example of how this type of binary consensus can evolve.
3.2 Binary consensus using three states

Figure 3.2: Binary Consensus using three states. Nodes start with colours white and black. Undecided nodes are denoted by grey colour (state $e$).

Now, let $X_1(z) = 1, X_2(z) = 0$ if node $z$ is in state 1 and $X_1(z) = 0, X_2(z) = 1$ if it is in state 2. Also, $X_1(z) = X_2(z) = 0$ if the node is undecided (state $e$).
Let $e_z$ be a vector of dimension $N$ (size of the network) with all coordinates equal to 0 except the $z$th one which is equal to 1. The state of the underlying continuous Markov chain, $(X_1, X_2)$, then changes to the following states with the following rates:

$$(X_1 + e_z, X_2) \text{ with rate } (1 - X_1(z) - X_2(z)) \sum_w P_{zw} X_1(w)$$

$$(X_1, X_2 + e_z) \text{ with rate } (1 - X_1(z) - X_2(z)) \sum_w P_{zw} X_2(w)$$

$$(X_1 - e_z, X_2) \text{ with rate } X_1(z) \sum_w P_{zw} X_2(w)$$

$$(X_1, X_2 - e_z) \text{ with rate } X_2(z) \sum_w P_{zw} X_1(w)$$

Consider the first line which corresponds to node $z$ changing its state to 1. Note that the term $(1 - X_1(z) - X_2(z))$ is only 1 when node $z$ is in state $e$ while $\sum_w P_{zw} X_1(w)$ denotes the sum of the probabilities of contacting neighbours in state 1.

Define $X_i = \sum_{z=1}^{N} X_i(z)$. In other words, $X_i$ will be the number of nodes which choose state $i$ (equivalently, these are the nodes at state $i$). As before, $P_{zw}$ is equal to $\frac{1}{N-1}$, $\forall z, w \in V$ for complete graphs if the probability distribution is uniform. The Markov process $(X_1, X_2)$ then evolves as follows:

$$(X_1, X_2) \rightarrow \begin{cases} 
(X_1 + 1, X_2) \text{ with rate } (N - X_1 - X_2) X_1/(N - 1) \\
(X_1 - 1, X_2) \text{ with rate } X_1 X_2/(N - 1) \\
(X_1, X_2 + 1) \text{ with rate } (N - X_1 - X_2) X_2/(N - 1) \\
(X_1, X_2 - 1) \text{ with rate } X_1 X_2/(N - 1) 
\end{cases} \quad (3.1)$$
3.2 Binary consensus using three states

Considering the first line which corresponds to the increment in the number of nodes in state 1, it can be seen that the rate is defined using the following terms; \((N - X_1 - X_2)\) which denotes the number of nodes in state \(e\), and \(X_1\) which denotes the number of nodes in state 1. In other words this is the rate at which nodes in state \(e\) contact the ones in state 1 and consequently change their state to 1.

The above Markov chain will terminate at one of the two states \((N, 0)\), or \((0, N)\) which correspond to the cases where all the nodes choose choices 1 or 2 as the initial majority respectively. Note that at any point in time \(X_1 + X_2 \leq N\).

### 3.2.2 Probability of error

To find the probability of error, let \(X^t_i\) and \(U^t\) denote the number of nodes in state \(i\) and \(e\) at time \(t\) respectively. Here, the initial number of nodes in states 1 and 2 are such that \(X^0_1 > X^0_2\) (i.e. the initial majority of nodes choose choice 1). Consider the following definition:

\[
g(X_i, X_j) \text{ is the probability of } (X^t_i, X^t_j) = (N, 0)
\text{ for some } t \geq 0 \text{ given that } (X^0_i, X^0_j) = (X_i, X_j)
\]

\(g(X_1, X_2)\) and \(g(X_2, X_1)\) are then the probability of all nodes ending in state 1 and 2 respectively (where the initial number of nodes in state 1 is \(X_1\) and the initial number of nodes in state 2 is \(X_2\)). Note that in the case of binary consensus where \(X^0_1 > X^0_2\), \(g(X_2, X_1) = 1 - g(X_1, X_2)\) is the probability of
3.2 Binary consensus using three states

error. Using this definition, the next step is to use the first step analysis\textsuperscript{2}, e.g. [33]. With many Markov chain problems, first step analysis can help with regards to finding the absorption probabilities by defining a recursive formula based on what happens after the first step of the process. Here, the following recursion is the direct result of using the first step analysis with (3.1).

\begin{equation}
(UX_1 + UX_2 + 2X_1X_2) g(X_1, X_2) = \\
UX_1 g(X_1 + 1, X_2) + UX_2 g(X_1, X_2 + 1) + \\
X_1X_2 g(X_1 - 1, X_2) + X_1X_2 g(X_1, X_2 - 1)
\end{equation} \tag{3.3}

where $U = N - X_1 - X_2$. Clearly, error occurs when the system hits $(0, N)$. Also, note that $g(X_i, X_i) = \frac{1}{2}$ because of the symmetry of the consensus process. In [1, Lemma 1], it is proved that the solution to (3.3) is given by,

\begin{equation}
g(X_1, X_2) = \frac{1}{2} g(X_1 - 1, X_2) + \frac{1}{2} g(X_1, X_2 - 1)
\end{equation} \tag{3.4}

with boundary conditions $g(0, X_2) = 0$ and $g(X_1, 0) = 1$.

Let $\alpha \in (\frac{1}{2}, 1]$ exist such that $\frac{(X_1^0, X_2^0)}{N} \rightarrow (\alpha, 1 - \alpha)$ when $N \rightarrow \infty$. It is then proved that for large $N$ the error probability $P_e$ (equivalent of $1 - g(X_1, X_2)$) decays exponentially with $N$ with the following rate,

\begin{equation}
\lim_{N \rightarrow \infty} \frac{1}{N} \log_2 P_e = -[1 - H(\alpha)]
\end{equation} \tag{3.5}

where $H(y) = -y \log_2(y) - (1 - y) \log_2(1 - y)$, for $y \in [0, 1]$. Note that the

\textsuperscript{2} For more information on first step analysis refer to Appendix A.
rate of decay depends on $\alpha$, the proportion of the initial majority.

### 3.2.3 Convergence time

To find the convergence time the authors in [1] use the following approach. Let $x_{i,N} = \frac{X_i}{N}$ ($i \in \{1, 2\}$) and $u_N = \frac{U}{N}$. Also, note that $X_1 + X_2 + U = N$. As the Markov process $(X_1, X_2, U)$ is a density dependent jump process the Kurtz’s theorem can be applied and the results of convergence in [34] can be used. The implication of Kurtz’s theorem with regards to the Markov process $(X_1, X_2, U)$ is then as follows (refer to Appendix B for an overview of Kurtz’s theorem). If $(x_{1,N}^0, x_{2,N}^0, u_N^0)$ converges to a fixed $(x_1^0, x_2^0, u^0)$ as $N$ goes to infinity, then $(x_{1,N}^t, x_{2,N}^t, u_N^t)$ converges to $(x_1^t, x_2^t, u^t)$ which satisfies the following set of differential equations (derived similar to (3.1)):

\[
\begin{align*}
\frac{dx_1^t}{dt} &= x_1^t u^t - x_1^t x_2^t, \\
\frac{dx_2^t}{dt} &= x_2^t u^t - x_2^t x_1^t, \\
\frac{du^t}{dt} &= 2x_1^t x_2^t - u^t(x_1^t + x_2^t)
\end{align*}
\]

Substituting $u^t$ with $1 - x_1^t - x_2^t$ yields:

\[
\begin{align*}
\frac{dx_1^t}{dt} &= x_1^t(1 - x_1^t - 2x_2^t), \\
\frac{dx_2^t}{dt} &= x_2^t(1 - x_2^t - 2x_1^t)
\end{align*}
\]

It is then proved in [1, Theorem 2] that for $x_1^0 > x_2^0$ the time $T$ to get to the
3.3 Binary Consensus using four states (binary interval consensus)

state \((x_1^t, x_2^t)\) such that \(1 - x_1^t\) and \(x_2^t\) are of order \(\frac{1}{N}\) is:

\[
T \sim \log N \quad (3.6)
\]

i.e., \(\lim_{N \to \infty} T/\log N = 1\).

Accordingly the convergence time for the binary consensus using an additional state is logarithmic in \(N\).

The results for the binary consensus process using three states show a significant improvement compared with the voter model which motivates us to apply the same approach to consensus on more than two choices.

3.3 Binary Consensus using four states (binary interval consensus)

We now go over the binary interval consensus which solves the binary consensus problem using four states for memory and communication. Binary Interval Consensus converges to the correct state with almost sure probability and therefore is of interest.

With the binary interval consensus each node can be in one of four states. In \([7]\) these states are denoted by 0, 0.5\(^-\), 0.5\(^+\), and 1 where \(0 < 0.5^- < 0.5^+ < 1\). Here, being in state 0 or 0.5\(^-\) means that a node believes the initial majority was 0 or equivalently the average values of the nodes is between 0.5 and 0.
3.3 Binary Consensus using four states (binary interval consensus)

3.3.1 Dynamics

Consider the setting in which node $j$ is contacted by $i$ with Poisson rate $q_{i\rightarrow j} = \frac{1}{2}P_{ij}$. Similarly, $q_{j\rightarrow i} = \frac{1}{2}P_{ji}$. In the set-up of [7], each pair of nodes interact at instances of a Poisson rate $q_{i,j}$ (where $q_{i,j} = q_{j,i} \neq 0$ if $(i, j) \in E$). For $i \neq j$, we then have

$$q_{ij} = q_{ji} = q_{i\rightarrow j} + q_{j\rightarrow i} = \frac{1}{2}(P_{ij} + P_{ji}) \quad (3.7)$$

In this chapter the probability matrix $P$ is considered a doubly stochastic matrix. This is a convenient assumption in the context of distributed algorithms (e.g. [27]) which enables us to use optimization techniques later in Chapter 5.

Accordingly, we define the rate matrix $Q$ as follows,

$$Q(i,j) = \begin{cases} 
q_{ii} = -\sum_{l \in V \setminus \{i\}} q_{il} & i = j \\
q_{ij} & i \neq j
\end{cases} \quad (3.8)$$

Note that when there is a self-loop for each node, $q_{ii}$ is not always -1. In other words $q_{ij}$ defines the activation rate of the edge $(i, j)$.

Let us also define the following relationship between $Q$ and a doubly stochastic matrix $P$,

$$Q = \frac{P + P^T}{2} - I_N \quad (3.9)$$

where $I_N$ is the identity matrix of size $N$. We use the matrix $\frac{P + P^T}{2}$ frequently
as an intermediary which relates $P$ and $Q$. We therefore denote $\frac{P + P^T}{2}$ by $P'$,

$$P' = \frac{P + P^T}{2}$$

which gives the following,

$$Q = P' - I_N$$

While the above relationship between $P$ and $Q$ is not needed in the rest of this section we will be using it later in Chapter 5 where we aim to optimize the binary interval consensus.

A sample of matrices $Q, P$ and $P'$ might clarify the mentioned equations more. Consider the following graph,

Figure 3.3: Simple graph with self-loops.
A double stochastic matrix $P$ for the above graph could be the following,

$$
P = \begin{pmatrix}
0.3670 & 0.4244 & 0.2087 & 0 \\
0.4293 & 0.1632 & 0.1590 & 0.2485 \\
0.2037 & 0.1671 & 0.1637 & 0.4655 \\
0 & 0.2453 & 0.4687 & 0.2860
\end{pmatrix}
$$  \hfill (3.12)

$$
P' = \begin{pmatrix}
0.3670 & 0.4269 & 0.2062 & 0 \\
0.4269 & 0.1632 & 0.1630 & 0.2469 \\
0.2062 & 0.1630 & 0.1637 & 0.4671 \\
0 & 0.2469 & 0.4671 & 0.2860
\end{pmatrix}
$$  \hfill (3.13)

$$
Q = \begin{pmatrix}
-0.9875 & 0.4962 & 0.4914 & 0 \\
0.4962 & -0.9924 & 0.0016 & 0.4945 \\
0.4914 & 0.0016 & -0.9985 & 0.5055 \\
0 & 0.4945 & 0.5055 & -1.0000
\end{pmatrix}
$$  \hfill (3.14)

Note that while $P$ is doubly stochastic it is not symmetric. However, $P'$ and subsequently $Q$ are both symmetric.

Now consider the interaction between any pair of nodes $(i, j)$. At each contact of the two nodes $i, j$ their states get updated using the following mapping:

If $i$ is 0 and $j$ is 1, they update to states $0.5^+$ and $0.5^-$,

If $i$ is $0.5^-$ and $j$ is 1, they update to states 1 and $0.5^+$,

If $i$ is $0.5^+$ and $j$ is 0, they update to states 0 and $0.5^-$,
3.3 Binary Consensus using four states (binary interval consensus)

If $i$ is $0.5^-$ and $j$ is 0, they update to states 0 and $0.5^-$,

If $i$ is $0.5^+$ and $j$ is 1, they update to states 1 and $0.5^+$,

If $i$ is $0.5^-$ and $j$ is $0.5^+$, they update to $0.5^+$ and $0.5^-$.

Using this mapping the binary interval consensus has the following properties [2]: Define $X_i$ as the state of node $i$. Following the interaction of nodes $i, j$ at time $t$,

- **Mixing:** It can be seen that if $X_i(t) \leq X_j(t)$ then $X_i(t+1) \geq X_j(t+1)$

- **Contraction:** $X_i(t + 1)$ and $X_j(t + 1)$ are either equal or one point away from each other (in the sequence $0, 0.5^-, 0.5^+, 1$).

- **Conservation:** Finally,

\[
X_i(t + 1) + X_j(t + 1) = X_i(t) + X_j(t) \quad (3.15)
\]

The last property means that the average is preserved throughout the consensus process. Note that $0.5^+$ and $0.5^-$ can both be considered as 0.5 when they are viewed as the value of the nodes. The only difference is that $0.5^+$ refers to the quantization level of $(0.5, 1]$ while $0.5^-$ refers to $[0, 0.5)$. We will use this property in Chapter 6 when we aim to define a framework for consensus in dynamic networks.

From the above mapping, it can be seen that the number of nodes in both states 1 and 0 will decrease by 1 only when a node in state 1 interacts with a
node in state 0. In any other case, they remain unchanged. In the following
the set of nodes in state $i$ at time $t$ will be denoted by $S_i(t)$. Assume that
nodes in state 0 are considered as the majority, and let $|S_0| = |S_0(0)| = \alpha N$
and $|S_1| = |S_1(0)| = (1 - \alpha)N$ where $\frac{1}{2} < \alpha \leq 1$. It can be seen that
throughout the consensus process the numbers of the nodes in state 1 and 0
only decrease at the encounters between 0 and 1. As a consequence, there
will eventually be no nodes in state 1 left in the network and the number of
nodes in state 0 will be $|S_0| - |S_1|$ at the end of the process. When all the
nodes with state 1 disappear from the network, there will be only nodes in
state $0^+, 0^-$, and 0 left. This phase of algorithm in which nodes in state 1
deplete from the network is called Phase 1 in the set-up of [7] (or “depletion
of 1s”).

Similarly, using the same mapping, it can be seen that the number of nodes
in state $0^+$ will decrease when they interact with nodes in state 0 and
consequently after some time the nodes in state $0^+$ will also disappear
from the network and only nodes in state 0 or $0^-$ will remain. At the end
of this stage where the algorithm actually reaches the consensus, the number
of nodes in state $0^-$ is $2(1 - \alpha)N$ and the number of nodes in state 0 is
$(2\alpha - 1)N$. This means that all the nodes agree that the average is on the
interval $[0,0.5)$ which indicates that nodes with state 0 were initially at the
majority. In [7], this phase of the algorithm is denoted by Phase 2. As
pointed out, at the end of phase 2 no node will be in $0^+$ (where nodes in
state 1 had already been depleted in phase 1), and because phase 2 starts
after phase 1, it can also be described as “depletion of $0^+$s”.

---

3.3 Binary Consensus using four states (binary interval consensus)
3.3 Binary Consensus using four states (binary interval consensus)

Accordingly, the process of the consensus can be considered as comprising two phases: Phase 1 where the nodes in state 1 disappear followed by Phase 2 where the nodes in state $0.5^+$ deplete from the network.

Note that throughout the consensus process the sum of the values of nodes always stays the same. For example if five nodes start with initial states $(0, 1, 0, 0, 1)$ in the end they will have the states $(0.5^-, 0.5^-, 0.5^-, 0.5^-, 0)$. While the result vector means that all the nodes agree that the average value is between zero and one and the initial majority is zero, the sum of the values always stays 2. A sample of such process can be seen in Figure 3.4.
3.3 Binary Consensus using four states (binary interval consensus)

Figure 3.4: Binary interval consensus. Nodes start with colours white and black. Colours light grey and dark grey denote states $0.5^-$, $0.5^+$ respectively.
3.3 Binary Consensus using four states (binary interval consensus)

3.3.2 Convergence time

In [7], the upper bounds for the expected time for each of these phases have been derived. These upper bounds are in terms of the eigenvalues of a set of matrices that depend on $Q$. If $S$ is considered as a non-empty subset of $V$, the set of vertices, $Q_S$ is defined as:

$$Q_S(i, j) = \begin{cases} 
-\sum_{l \in V} q_{il} & i = j \\
q_{ij} & i \notin S, j \neq i \\
0 & i \in S, j \neq i 
\end{cases} \quad (3.16)$$

It can be seen that $Q_S$ not only depends on the structure of the graph but also depends on $\alpha$. The following lemma is then derived:

**Lemma 2.** For any finite graph $G$, there exists $\delta(G, \alpha) > 0$ such that, for any non-empty subset of vertices $S$ ($|S| < N$), if $\lambda_1(Q_S)$ is the largest eigenvalue of $Q_S$, then it satisfies

$$\delta(G, \alpha) = \min_{S \subseteq V, |S| = (2\alpha - 1)N} |\lambda_1(Q_S)| \quad (3.17)$$

Note that using this definition, for all non-empty set $S$, $\delta(G, \alpha) > 0$ because $\lambda_1(Q_S) < 0$. The upper bounds for Phase 1 and 2 are then found:

**Theorem 3.** If $T_1$ is considered as the smallest time it takes for the nodes in state 1 to deplete from the network, then the expected value of $T_1$ is bounded
3.3 Binary Consensus using four states (binary interval consensus)

as follows,

\[ \mathbb{E}(T_1) \leq \frac{1}{\delta(G, \alpha)} \left( \log N + 1 \right). \quad (3.18) \]

Now, if \( T_2 \) is considered as the time it takes for the nodes in state 0.5+ to deplete starting from a state without any nodes in state 1, then

\[ \mathbb{E}(T_2) \leq \frac{1}{\delta(G, \alpha)} \left( \log N + 1 \right). \quad (3.19) \]

And consequently, considering both (3.18) and (3.19), if \( T \) is considered as the time of convergence (i.e. the time it takes for nodes in states 1 and 0.5+ to deplete), it is bounded as,

\[ \mathbb{E}(T) \leq \frac{2}{\delta(G, \alpha)} \left( \log N + 1 \right). \quad (3.20) \]

Theorem 3 shows that the bound on the time of convergence depends on the voting margin \( \alpha \) and the structure of the graph.

Considering (3.20), it can be seen that the time of convergence directly depends on \( \delta(G, \alpha) \). Accordingly, in order to reduce the upper bound of the convergence time, \( \delta(G, \alpha) \) should be maximized. Considering the definition of \( \delta(G, \alpha) \), this can be achieved by minimizing the eigenvalues of the \( Q_S \) matrix. However, it is very difficult to optimize the eigenvalues of the \( Q_S \) matrix directly as they very much depend on the dynamics of the algorithm itself.

In Chapter 5, we try to find the bounds of \( \delta(G, \alpha) \) in terms of the eigenvalues of the \( Q \) matrix which depends on \( P' \) (through (3.11)). Let us now introduce the eigenvalue optimization problems that can be related to our optimization
3.4 Fastest mixing Markov chain problem

In this section we give a brief overview of the fastest mixing Markov chain (FMMC) problem which has been investigated in [28].

3.4.1 Markov Chain on an undirected graph $G(V,E)$

As before, consider an undirected graph $G(V,E)$. A Markov Chain $X(t)$ of graph $G$ is defined as follows,

$$
\hat{P}_{ij} = P(X(t+1) = j | X(t) = i), \forall i,j = 1,\ldots,N
$$

In this setting the $\hat{P}_{ij}$s define the state transition probabilities of the Markov chain. Concretely, each $\hat{P}_{ij}$ corresponds to the probability of a transition from node $i$ to $j$ (i.e. transition from state $i$ to state $j$). For each $\forall (i,j) \notin E, \hat{P}_{ij} = 0$.

We also require the probability matrix $\hat{P}$ to satisfy the following conditions.

$$
\hat{P} = \hat{P}^T, \hat{P} \geq 0, \hat{P} \mathbf{1} = \mathbf{1}
$$

where $\mathbf{1}$ is a vector of all ones. Note that $\geq$ is an element-wise operator meaning that $\hat{P}_{ij} \geq 0, \forall i,j$. The first condition shows the symmetry of the
probability matrix (the graph $G$ is undirected) and the second and third conditions are the inherent properties of a doubly stochastic probability matrix. Also, note that self-loops are allowed.

Let $\mu(t)$ be the distribution of the states of the Markov chain at time $t$. Accordingly, $\mu_i(t) = \mathbb{P}(X(t) = i)$ and

$$\mu(t + 1)^T = \mu(t)^T \hat{P}$$

Since $1^T \hat{P} = 1^T$, it is not difficult to see that the uniform state distribution, $\mu = (1/N)1$, is the equilibrium state distribution.

### 3.4.2 Fastest mixing problem

Convergence rate of $\mu$ to the uniform distribution depends on the Second Largest Eigenvalue Modulus (SLEM) of the $P$ matrix, $\zeta$, defined as follows:

$$\zeta(\hat{P}) = \max\{\lambda_2(\hat{P}), -\lambda_n(\hat{P})\}$$

where $1 = \lambda_1(\hat{P}) \geq \ldots \geq \lambda_n(\hat{P})$ are the eigenvalues of $\hat{P}$. $\zeta$ is also known as the mixing rate of the Markov chain. It is stated in [28] that when $\zeta < 1$, $X(t)$ converges to the uniform distribution asymptotically as $\zeta^t$ ($t \to \infty$). This means that the smaller the $\zeta$, the faster the convergence will be. In particular, the Fastest Mixing Markov Chain problem is defined as follows.
minimize $\zeta(\hat{P})$
subject to $\hat{P}_{ij} \geq 0$, $\hat{P}_{ij} = 0$ if $i, j \notin E$
and $\sum_j \hat{P}_{ij} = 1, \forall i$ \hfill (3.21)

The FMMC corresponds to finding the symmetric probability matrix $\hat{P}$ that
gives the fastest Markov chain. Note that $\hat{P}$ is symmetric.

### 3.4.3 Convexity of $\zeta(\hat{P})$

It can then be proved that the FMMC is a convex optimization problem. One
way to show this is to use the variational characterisations of eigenvalues \cite{35}.

The following is part of the Rayleigh-Ritz theorem.

**Theorem 4.** The largest eigenvalue of a Hermitian matrix $H$ can be derived
as a solution of the following optimization problem (Note that $\|\cdot\|_2$ denotes
the matrix norm with respect to the Euclidean norm),

$$\lambda_1(H) = \sup \{ v^* Hv \mid \|v\|_2 = 1 \}$$ \hfill (3.22)

**Proof:** $H$ is Hermitian and therefore can be decomposed as $QAQ^*$ where $Q$
is a unitary matrix and $\Lambda$ is a diagonal matrix with the eigenvalues of $H$ as
3.4 Fastest mixing Markov chain problem

its diagonal elements. For any \( v \) vector in \( \mathbb{R}^N \) we then have the following,

\[
v^* M v = v^* Q \Lambda Q^* v = \sum_{i=1}^{N} \lambda_i |(Q^* v)_i|^2
\]  

(3.23)

As \( |(Q^* v)_i|^2 \geq 0 \) the right hand side of (3.23) is less than \( \lambda_1 \sum_{i=1}^{N} |(Q^* v)_i|^2 \).

Also,

\[
\lambda_1 \sum_{i=1}^{N} |(Q^* v)_i|^2 = \lambda_1 \sum_{i=1}^{N} |v_i|^2 = \lambda_1 v^* v
\]

(3.24)

This concludes the proof.

We know that the largest eigenvalue of the matrix \( \hat{P} \) is 1 with the corresponding eigenvector \( \mathbf{1} \) and therefore we can express \( \lambda_2 \) by using vectors \( u \) which are orthogonal to the eigenvector \( \mathbf{1} \). The second largest eigenvalue is then defined as follows:

\[
\lambda_2(\hat{P}) = \sup\{u^T \hat{P} u \mid \|u\|_2 = 1, \mathbf{1}^T u = 0\}
\]

(3.25)

\( u^T \hat{P} u \) is a set of linear functions of \( \hat{P} \) and \( \lambda_2(\hat{P}) \) is a point-wise supremum of \( u^T \hat{P} u \). Therefore, \( \lambda_2(\hat{P}) \) is a convex function of \( \hat{P} \). We also can express \( -\lambda_n(\hat{P}) \) as follows:

\[
-\lambda_n(\hat{P}) = \sup\{-u^T \hat{P} u \mid \|u\|_2 = 1\}
\]

(3.26)

Consequently both \( \lambda_2(\hat{P}) \) and \( \lambda_n(\hat{P}) \) are convex functions and therefore \( \zeta(\hat{P}) \) is also a convex function. The FMMC problem can then be considered as a convex optimization problem.
The convexity of $\zeta(\hat{P})$ can also be shown by expressing it as the norm of $\hat{P}$ restricted to the subspace $1^\perp$ (since the eigenvector associated with $\lambda_1$ is 1) \cite{28}.

\begin{equation}
\zeta(\hat{P}) = \| (I - (\frac{1}{N})11^T)\hat{P} (I - (\frac{1}{N})11^T) \|_2 \\
= \| \hat{P} - (\frac{1}{N})11^T \|_2 \tag{3.27}
\end{equation}

By expressing $\zeta(\hat{P})$ as a norm of $\hat{P}$ restricted to the subspace $1^\perp$ the optimization problem (3.21) becomes a convex optimization problem.

3.4.4 SDP formulation

Concretely, by adding a scalar $c$ we can formulate the convex optimization problem as a Semi-Definite Programming (SDP) problem as follows:

\begin{align*}
\text{minimize} & \quad c \\
\text{subject to} & \quad -cI \preceq \hat{P} - (\frac{1}{N})11^T \preceq cI \\
& \quad \hat{P}_{ij} \geq 0, \quad \hat{P}_{ij} = 0 \text{ if } i, j \notin E \\
& \quad \text{and} \quad \sum_j \hat{P}_{ij} = 1, \forall i \tag{3.28}
\end{align*}

Here, $A \preceq B$ means that $A - B$ is positive semi-definite (PSD). Note that matrix $Y$ is positive semi-definite if and only if all the eigenvalues of $Y$ are non-negative.
3.4 Fastest mixing Markov chain problem

3.4.5 Sub-gradient of $\zeta(\hat{P})$

It is pointed out that there are primal-dual interior point algorithms which can solve this SDP problem. They include the ones in [36], [37] and [38]. However, for large graphs this problem can be solved using a sub-gradient method which uses the sparsity of $\hat{P}$. Sub-gradient methods are widely used to solve eigenvalue optimization problems (e.g. [39], [40], and [41]). More general background on non-smooth optimization can be found in [42], [43], and [44].

A sub-gradient of $\zeta$ at $\hat{P}$ is a symmetric matrix, $S$, such that:

$$\zeta(\tilde{P}) \geq \zeta(\hat{P}) + \sum_{i,j} S_{ij}(\tilde{P}_{ij} - \hat{P}_{ij})$$

(3.29)

for any symmetric and stochastic matrix $\tilde{P}$. The following corollary is then true [28].

**Corollary 5.** If $u$ is a unit eigenvector associated with $\lambda_2(\hat{P})$ and $\zeta(\hat{P}) = \lambda_2(\hat{P})$, the matrix $S = uu^T$ is a sub-gradient of $\zeta(\hat{P})$.

**Proof.** We know that $u^T \mathbf{1} = 0$. Furthermore by the variational characterization of the second eigenvalue of $\hat{P}$ and $\tilde{P}$,

$$\zeta(\tilde{P}) \geq \lambda_2(\hat{P}) \geq u^T \tilde{P} u$$

Subtracting the two sides of the second equation from those of the first one
3.4 Fastest mixing Markov chain problem

results in the following which is required:

\[
\zeta(\tilde{P}) \geq \zeta(\hat{P}) + u^T(\tilde{P} - \hat{P})u = \zeta(\hat{P}) + \sum u_iu_j(\hat{P}_{ij} - \hat{P}_{ij}) \tag{3.30}
\]

Repeating the same for the case of \(\zeta(\hat{P}) = -\lambda_n(P)\) the matrix \(-uu^T\) will be the sub-gradient of \(\zeta(\hat{P})\).

The authors in [28] then use a Projected Sub-gradient Method to optimize the \(\hat{P}_{ij}\)s in order to minimize the \(\zeta(\hat{P})\).

3.4.6 Optimization

To optimize \(\zeta(\hat{P})\), first the non-diagonal elements of the \(\hat{P}\) matrix (i.e. \(\hat{P}_{ij}\), \(i \neq j\)) are passed to a vector \(\psi\). As each \(\hat{P}_{ij}\) corresponds to the edge \((i,j)\) we can label these edges by \(r = 1, \ldots, m\) and write \(\hat{P}\) as an affine function of \(\psi\) as follows:

\[
\hat{P}(\psi) = I + \sum_{r=1}^{m} \psi_r B_r \tag{3.31}
\]

where \(B_r\) is an \(N \times N\) matrix and if edge \(r\) corresponds to \((i,j)\); \(B_{ri} = B_{ij} = -1, B_{rj} = B_{ji} = 1\) and zero elsewhere. Note that we can calculate the probabilities of staying at nodes (i.e. \(\hat{P}_{ii}\)) using the fact that \(\hat{P}\mathbf{1} = \mathbf{1}\).

Define matrix \(D\) of size \(N \times m\) as follows, \(D_r = 1\) if an edge \(r\) is incident to node \(i\) and zero elsewhere. The diagonal of \(\hat{P}\) which is the vector of the probabilities of staying at the nodes is then equal to \(I - D\psi\). \(D\) is defined to make sure that the sum of the elements in each row is 1 (since the graph is simple). Concretely any feasible vector \(\psi\) should satisfy the following
3.4 Fastest mixing Markov chain problem

constraints:

\[ \psi \geq 0, D\psi \leq 1 \]  

(3.32)

Note that the inequalities in (3.32) are element-wise. The optimization problem can then be formulated as follows with the optimization problem \( \psi \),

\[
\begin{align*}
\text{minimize} & \quad \zeta(\hat{P}(\psi)) \\
\text{subject to} & \quad \psi \geq 0, D\psi \leq 1
\end{align*}
\]

(3.33)

The find the sub-gradient when \( \zeta(\hat{P}(\psi)) = \lambda_2(\hat{P}(\psi)) \) we need to re-write (3.30) (using (3.31)) as follows,

\[
\zeta(\hat{P}(\tilde{\psi})) \geq \zeta(\hat{P}(\psi)) + \sum_{r=1}^{m} v^T B_r v(\tilde{\psi}_r - \psi_r)
\]

(3.34)

Therefore, for \( \zeta(\hat{P}(\psi)) = \lambda_2(\hat{P}(\psi)) \), sub-gradient \( f(\psi) \) is given by,

\[
f(\psi) = (v^T B_1 v, \ldots, v^T B_m v)
\]

(3.35)

where \( v \) is the unit eigenvector associated with \( \lambda_2(\hat{P}(\psi)) \) and its elements are,

\[
f_r(\psi) = v^T B_r v = (v_i - v_j)^2, r \sim (i, j), r = 1, \ldots, m
\]

(3.36)

Similarly, for the case of \( \zeta(\hat{P}(\psi)) = -\lambda_n(\hat{P}(\psi)) \) the sub-gradient \( f(\psi) \) is given by,

\[
f(\psi) = (-u^T B_1 u, \ldots, -u^T B_m u)
\]

(3.37)
3.4 Fastest mixing Markov chain problem

where \( u \) is the unit eigenvector associated with \(-\lambda_n(\hat{P}(\psi))\). The elements of \( f(\psi) \) can be written as,

\[
f_r(\psi) = -u^T B_r u = (u_i - u_j)^2, \quad r \sim (i, j), \quad r = 1, \ldots, m \tag{3.38}
\]

The authors in \[28\] propose using Lanczos methods to calculate the eigenvalues and their corresponding eigenvectors \([45\) and \(46\)].

Algorithm \[1\] shows the optimization steps.

Algorithm 1 Optimization

1: \( k \leftarrow 1 \)
2: repeat
3: // Subgradient Step
4: Calculate \( f^{(k)} \) and update \( \psi, \psi \leftarrow \psi - \beta_k f^{(k)} / \| f^{(k)} \|_2 \)
5: // Sequential Projection
6: \( \psi_r \leftarrow \max \{ \psi_r, 0 \}, \quad r = 1, \ldots, m \)
7: for each node \( i = 1, \ldots, N, \quad \mathcal{H}(i) = \{ r | \text{edge } r \text{ connected to } i \} \) do
8: while \( \sum_{r \in \mathcal{H}(i)} \psi_r > 1 \) do
9: \( \mathcal{H}(i) \leftarrow \{ r | r \in \mathcal{L}(i), \psi_r > 0 \} \)
10: \( \gamma \leftarrow \min \left\{ \min_{r \in \mathcal{H}(i)} \psi_r, \left( \sum_{r \in \mathcal{H}(i)} \psi_r - 1 \right) / |\mathcal{H}(i)| \right\} \)
11: \( \psi_r \leftarrow \psi_r - \gamma, \quad r \in \mathcal{H}(i) \)
12: end while
13: end for
14: \( k \leftarrow k + 1 \)

A brief description is as follows; The sub-gradient step in the algorithm moves \( \psi \) towards the sub-gradient with a step size which diminishes with \( k \) \((\beta_k \to 0, \sum \beta_k = \infty, \beta_k \geq 0)\).

The projection step aims to project \( \psi \) to a feasible set (i.e. a set that satisfies \((3.32)\), in particular \( D\psi \leq 1 \)). This is implemented using a sequential projection method in which first \( \psi \) is projected on a non-negative orthant and then
3.4 Fastest mixing Markov chain problem

projected on one half-space at a time. Here, $\mathcal{H}(i)$ in each loop denotes the set of edges connected to vertex $i$ and with strictly positive transition probabilities. Denote the sum of the $\psi_i$s for node $i$ by $S_\psi$ (i.e. $S_\psi = \sum_{r \in \mathcal{H}(i)} \psi_r$). If $S_\psi \geq 1$ then $\psi$ should be projected to the half space $\sum_{r \in \mathcal{H}(i)} \psi_r \leq 1$. However this projection must not make any of the elements of $\psi$ negative and therefore the $\psi$ is projected to the following half space,

$$\sum_{r \in \mathcal{H}(i)} \psi_r \leq S_\psi - \gamma |\mathcal{H}(i)| \quad (3.39)$$

Here $\gamma$ is chosen such that the elements of the projection will not become negative. Define $\psi^*$ as the optimal solution of the minimization problem, $\psi$ the probability vector after the sub-gradient step and $\bar{\psi}$ the probability vector after the sequential projection step, then the following inequality holds [28],

$$\|\psi^* - \bar{\psi}\|_2 \leq \|\psi^* - \psi\|_2 \quad (3.40)$$

This means that the distance between the vector $\psi$ and the optimal solution $\psi^*$ does not increase after each step. Consequently, (3.40) can prove the convergence of the optimization algorithm (e.g. [47], [48]).
3.5 Distributed averaging and optimization of convergence time

Distributed averaging is one of the most straightforward consensus algorithms in networks. There have been numerous analysis on the averaging algorithm (e.g. [49], [50], [51], and [52]). Distributed averaging is reliable and always converges to the correct result (as long as the graph is connected), however, it needs relatively high amount of memory and processing resources. A simple application of an averaging algorithm can be finding a better estimate of a certain variable (e.g. the temperature of a room) by averaging the measured value of multiple nodes (e.g. wireless sensors). We review the results in [27] regarding the algorithm, the bounds on its convergence time and the distributed optimization techniques that were used to optimize its convergence time.

3.5.1 Algorithm and dynamics

The averaging algorithm in [27] is a distributed process defined as follows. Initially each node has an assigned (or measured) value and the goal for all the nodes is to find the average value of the values of all the nodes in a distributed fashion. To this end, at each time step one of the nodes’ clocks ticks (with Poisson rate 1). It then contacts one of its neighbours using a probability defined by the doubly stochastic matrix $P$. This is equivalent of a global clock which ticks with Poisson rate $N$. Every time this global clock ticks a pair of nodes (which are neighbours) interact with each other. After
3.5 Distributed averaging and optimization of convergence time

the interaction between the two nodes they set their values to the average of their previous values and this goes on until all the nodes converge to the same value which is the desired result.

Define $s(k)$ as the vector of the states of the nodes at time step $k$ where $s_i(k)$ corresponds to the value of node $i$. $s$ then evolves as follows:

$$s(k) = M(k)s(k - 1) \quad (3.41)$$

where $M(k)$ is a random matrix and if at time step $k$ nodes $i, j$ interact is given by,

$$M_{ij} = I - (c_i - c_j)(c_i - c_j)^T/2 \quad (3.42)$$

Here, $c_i$ is a vector of size $N$ with the $i$th component equal to 1 and zero elsewhere. Also, the probability of the interaction between nodes $i, j$ at time step $k$ is $\frac{1}{N}P_{ij}$ in the asynchronous setting.

3.5.2 Convergence time

Before giving the bounds of the convergence time of the averaging algorithm we need to define a measure of closeness to the result. This is called $\epsilon$-averaging time and is expressed as follows; The smallest time that it takes for $s(t)$ to converge to the $\epsilon$ proximity of $\hat{s}1$ with high probability starting from any $s(0)$ (where $\hat{s}$ is the average of all the values of $s$),

$$\hat{i}(P, \epsilon) = \max_{s(0)} \min_{t : P(\|s(t) - \hat{s}1\| \geq \epsilon) \leq \epsilon} t$$

(3.43)
3.5 Distributed averaging and optimization of convergence time

The following theorem gives the bounds for convergence time as stated in [27].

**Theorem 6.** The bounds of the convergence time, \( \hat{t}(P, \epsilon) \) are given by,

\[
0.5 \log \frac{1}{\epsilon} \leq \hat{t}(P, \epsilon) \leq 3 \log \frac{1}{\epsilon}
\]

where

\[
M = \frac{1}{N} P_{ij} M_{ij} = I - \frac{1}{2N} F + \frac{P + P^T}{2N}
\]

Here \( F \) is a diagonal matrix with elements \( F_i = \sum_{j=1}^{N} [P_{ij} + P_{ji}] \) and \( \lambda_2(M) \) denotes the second largest eigenvalue of \( M \).

Note that these bounds are in terms of the number of time steps (ticks) and not the absolute time. The absolute time can be derived using Lemma 1 in Section 2.3.

### 3.5.3 Optimization

It is clear from Theorem 6 that the convergence time has an inverse relationship with \( \lambda_2(M) \). Accordingly to optimize the convergence time \( \lambda_2(M) \) must be reduced. This results in the following optimization problem:

\[3\lambda_1(M) \geq \ldots \geq \lambda_n(M) \text{ are the eigenvalues of } M\]
3.5 Distributed averaging and optimization of convergence time

minimize $\lambda_2(M)$

subject to $P_{ij} \geq 0, P_{ij} = 0$ if $i, j \notin E,$

$$\sum_j P_{ij} = 1, \forall i$$ (3.45)

Note the similarities between the FMMC problem (3.21) in Section 3.4 and (3.45). Similar to the FMMC problem the objective function is a convex function (the second largest eigenvalue of a symmetric and doubly stochastic matrix) and can be solved using SDP. The formulation then becomes the following,

minimize $c$

subject to $M - \frac{11^T}{N} \preceq cI$

$M = \frac{1}{N} P_{ij} M_{ij}$

$P_{ij} \geq 0, P_{ij} = 0$ if $i, j \notin E,$

$$\sum_j P_{ij} = 1, \forall i$$ (3.46)

By formulating the convex optimization problem as a SDP a sub-gradient method can be used to solve the problem. Similar to the FMMC problem the probabilities $P_{ij}$s are passed on to a vector $\psi'$ and the sub-gradient is $S' = zz^T$ where $z$ is the unit eigenvector associated with $\lambda_2(M)$ (similar to
3.5 Distributed averaging and optimization of convergence time

Corollary [5]. This time the objective function can be re-formulated as,

\[ M(\psi') = I + \frac{1}{2N} \left( \sum_{r=1}^{m} \psi'_r B_r + \psi'_{-r} B_{-r} \right) \]  (3.47)

Here, similar to the FMMC problem, \( m \) denotes the number of edges and each \( r \) denotes an edge \((r \sim (i,j))\). We need to define \( \psi'_{r} \) as well. This is because \( P_{ij,s} \) and \( P_{ji,s} \) are not necessarily the same. If \( r \sim (i,j) \), then \( B_{rij} = B_{rji} = 1 \) and \( B_{rii} = B_{rjj} = -1 \). Furthermore, the sub-gradient \( S' \) can be expressed as follows,

\[ f'(\psi') = \frac{1}{2N} \left( z^T B_{-m} z, \ldots, z^T B_{m} z \right) \]  (3.48)

Note the similarities between (3.37), (3.35), and (3.48). Accordingly the elements of the \( f'(\psi') \) are given by

\[ f'_r(\psi') = -\frac{1}{2N} (z_i - z_j)^2 \]  (3.49)

The sub-gradient method used in [27] to optimize \( \lambda_2(M) \) comprises of the Subgradient and Projection steps and is very similar to the one we discussed in the previous section for the FMMC problem and therefore we skip its description to avoid redundancy.

\footnote{Note that the \( P \) matrix in the distributed averaging is doubly stochastic but not necessarily symmetric while the \( \hat{P} \) matrix in the FMMC problem is both doubly stochastic and symmetric.}
3.5 Distributed averaging and optimization of convergence time

3.5.4 Decentralization

Equations (3.48) and (3.49) show that if each node of the network is aware of its corresponding entry and those of its neighbours in the eigenvector \( z \) the sub-gradient can be computed in a distributed way. The projection step which is very similar to that of the FMMC problem can also be computed at each node and using only local information.

This can only happen if there is a way to compute the eigenvector \( z \) in a distributed way and the authors in [27] suggest using the DECENTRAL OI (Decentralized Orthogonal Iterations) algorithm of [53]. They state that even by using an approximation of the eigenvector \( z \) the optimization variable converges to a close proximity of the optimal value.

Inspired by the approach in [27] we suggest a framework at the end of Chapter 5 to decentralize the optimization of the binary interval consensus.
Chapter 4

Multi-valued consensus

In this chapter we introduce an algorithm for the multi-valued consensus process (for $k$ choices) based on the distributed binary consensus algorithm in [1]. It is worth noting that a similar algorithm exists for multivalued consensus problem in [54] as part of the proposed algorithm for addressing the ranking problem. The so called plurality selection algorithm uses $2k$ states of memory and signalling and addresses the same majority consensus problem as ours in a complete graph. However, in our algorithm we use $k+1$ states for the communication and memory states as we are addressing the majority consensus problem and not the ranking problem. Also, our results and proofs for the convergence time and the probability of error differ from those of [54].

We reviewed the results for binary consensus in [1] in Section 3.2. This should help the reader to gain a better understanding of our approach towards solving the multivalued consensus problem. We show how we expand the
4.1 Multivalued Consensus

Here we present our algorithm which extends the binary consensus in [1] to a multivalued consensus on \( k \) choices. We then analyse its probability of error and convergence time.

4.1.1 Dynamics

As before, the goal for each node is to find the initial majority in a distributed fashion and also using the minimum communication and memory bits. Only this time the agreement is going to be on one of the \( k \) choices (states) as opposed to only two.

We use the same asynchronous set-up of Section 3.2. Each node has a clock which ticks with Poisson rate 1. At each time step, one of the nodes’ (node \( z \)) clocks ticks and it contacts a neighbour \( w \) with probability \( P_{zw} \). Similar to Section 3.2 if we consider the complete graph and a uniform probability distribution for the interactions between each node and its neighbours, the \( P_{zw} \)s are equal to \( \frac{1}{N-1} \) and node \( z \) will change its state according to the
message received from node \( w \).

Now, with multivalued consensus nodes can have one of the states \( 1, \ldots, k \) and \( e \) at any time, where \( e \) is the same additional state which shows that the node is undecided. In this case, when a node in state \( i \) contacts a node in state \( i \) or \( e \), it does not change its state and when it contacts a node in state \( j \), \( (i \neq j) \), it updates its state to \( e \). As before, if a node in \( e \) contacts a node in state \( i \), it changes its state to \( i \).

Similar to the binary case if a node \( z \) is in state \( i \) \( (i \in \{1, \ldots, k\}) \), \( X_i(z) = 1, X_j(z) = 0, \forall j \in \{1, \ldots, k\}\setminus i \). Also, state \( e \) is denoted by \( X_i(z) = 0, \forall i, 1 \leq i \leq k \). If node \( z \) contacts node \( w \) with probability \( P_{zw} \), the state of the Markov chain \((X_1, \cdots, X_k)\) of the graph evolves as follows:

\[
(X_1, \ldots, X_k) \rightarrow \begin{cases} 
(X_1 + e_z, \ldots, X_k), \text{ at rate } & (1 - X_1(z) - \ldots - X_k(z)) \sum_w P_{zw} X_1(w) \\
(X_1 - e_z, \ldots, X_k), \text{ at rate } & X_1(z) \sum_w P_{zw} (X_2(w) + \ldots + X_k(w)) \\
& \vdots \\
(X_1, \ldots, X_k + e_z), \text{ at rate } & (1 - X_1(z) - \ldots - X_k(z)) \sum_w P_{zw} X_k(w) \\
(X_1, \ldots, X_k - e_z), \text{ at rate } & X_k(z) \sum_w P_{zw} (X_1(w) + \ldots + X_{k-1}(w)),
\end{cases}
\]

where \( e_z \) as before, is a vector of dimension \( N \) (size of the network) with all coordinates equal to 0 except the \( z \)th coordinate which is equal to 1. The evolution of the Markov Chain represents the fact that by contacting node
4.1 Multivalued Consensus

$w$, the change of $X_i(z)$ from 0 to 1 only happens if node $w$ is in state $i$ (i.e. $X_i(w) = 1$) while node $z$ had been in state $e$. On the other hand the change of $X_i(z)$ from 1 to 0 happens when node $w$ is in any state other than $i$ and $e$, while node $z$ had been in state $i$.

Let $X_i = \sum_{z=1}^{N} X_i(z)$. Considering the complete graph, $P_{zw} = 1/(N - 1), \forall z, w \in V$ and the rates for the Markov process will then be the following:

\[
\begin{align*}
(X_1, \ldots, X_k) &\rightarrow \\
(X_1 + 1, \ldots, X_k) : \frac{(N - X_1 - \ldots - X_k)X_1}{(N - 1)}, \\
(X_1 - 1, \ldots, X_k) : \frac{X_1 (X_2 + \ldots + X_k)}{(N - 1)}, \\
& \vdots \\
(X_1, \ldots, X_k + 1) : \frac{(N - X_1 - \ldots - X_k)X_k}{(N - 1)}, \\
(X_1, \ldots, X_k - 1) : \frac{X_k (X_1 + \ldots + X_{k-1})}{(N - 1)}. 
\end{align*}
\]

Note that $\sum_{i=1}^{k} X_i \leq N$ and the Markov process terminates at one of the following $k$ states,

\[(N, 0, \ldots, 0), (0, N, \ldots, 0) \ldots, (0, \ldots, 0, N).\]

4.1.2 Probability of Error

To find the probability of error we first assume, without loss of generality, that the initial number of the nodes in states 1, 2, $\ldots$, $k$ are such that $X_1^0 > X_2^0 > \ldots > X_{k-1}^0 > X_k^0$. This means that the number of the initial votes for different choices are not the same and therefore there exists a choice with the
4.1 Multivalued Consensus

highest number of votes. Consider the following definition,

\[ f(X_1, \ldots, X_k) \text{ is the probability of } (X_1^t, \ldots, X_k^t) = (N, 0, \ldots, 0) \]

, for some \( t \geq 0 \), given that \((X_1^0, \ldots, X_k^0) = (X_1, \ldots, X_k) \) \hspace{1cm} (4.1)

\( f(X_1, \ldots, X_k) \) is then the probability of all the nodes ending in state 1 (where the initial number of nodes in states 1, \ldots, \( k \) is \( X_1, \ldots, X_k \) respectively).

Consider the following notation:

\[ f(X) = f(X_1, \ldots, X_k) \]

Also, define \( f_{X_l}(X) \) as the probability of all nodes ending in state \( l \) i.e.

\[ f_{X_l}(X) \text{ is the probability of } (X_1^t, \ldots, X_k^t) = (0, \ldots, \underbrace{N}_{l\text{th element}}, \ldots, 0) \]

, for some \( t \geq 0 \), given that \((X_1^0, \ldots, X_k^0) = (X_1, \ldots, X_k) \)

Also,

\[ f(X \pm e_i) = f(X_1, \ldots, X_{i-1}, X_i \pm 1, X_{i+1}, \ldots, X_k) \]

Using the first step analysis for state 1 we can derive the following equation,

\[ \left( U \sum_{i=1}^{k} X_i + 2 \sum_{i,j=1, i \neq j}^{k} X_i X_j \right) f(X) \]

\[ = U \sum_{i=1}^{k} X_i f(X + e_i) + \sum_{j=1}^{k} X_j \sum_{i=1, i \neq j}^{k} X_i f(X - e_j) \] \hspace{1cm} (4.2)
4.1 Multivalued Consensus

where $U = N - \sum_{i=1}^{k} X_i$. Accordingly, $f(X_1, 0, \ldots, 0) = 1$ and $f(0, X_2, \ldots, X_k) = 0$. Here, if $X_1^0 > \ldots > X_k^0$ the error occurs when the system hits any of the final states except $(N, \ldots, 0)$. Also, $f(X_i, \ldots, X_i) = \frac{1}{k}$ because of the symmetry of the protocol.

Finding the probability of error through solving (4.2) is not straightforward. Instead, we try to use the result for $g(X_1, X_2)$ in Section 3.2 to find an upper bound for the probability of error (i.e. $1 - f(X)$). We use the following Lemma.

**Lemma 7.** For two different consensus algorithms (binary and multivalued) applied on the same set of nodes, the following relationship, between the probability of ending in one of the two states in the binary consensus (3.2) and the probability of ending in one of the $k$ states in the multivalued consensus (4.1), holds.

$$f_{X_l}(X) \leq g(X_l, X_j), j \neq l$$

(4.3)

where $\sum_{i=1}^{k} X_i^0 = N$.

Consider two separate processes one a binary consensus and one a multivalued consensus which start from the same network (of size $N$). For the case of binary consensus $g(X_l, X_j)$ there are only two choices at the start with populations $X_l^0 = X_l$ and $X_j^0 = X_j$ (the rest of the nodes are undecided). For the case of multivalued consensus there are $k$ choices at the beginning and each node decides on one of these choices ($X_1^0 = X_1, \ldots, X_k^0 = X_k$). Lemma 7 follows by the fact that the probability of ending up in any of the absorbing states decreases when we have more choices. Consider the rates of increasing and decreasing of the number of nodes in state $l$. The rate of $X_l$ increasing
4.1 Multivalued Consensus

to $X_l + 1$ goes down by adding more choices; $(N - \sum_{i=1}^{k} X_i)X_l/(N - 1)$ compared with $(N - X_l - X_j)X_l/(N - 1)$. Furthermore, the rate of decrease to $X_l - 1$ goes up by adding extra choices; $X_l(\sum_{i\neq l, i=1}^{k} X_i)/(N - 1)$ compared with $X_lX_j/(N - 1)$.

Note that with $k$ choices when $(X_1^0 > \ldots > X_k^0)$, error occurs when the consensus finishes in any of the absorbing states other than $(N, \ldots, 0)$. This means that the new probability of error ($P_e$) can be defined as follows:

$$P_e = \sum_{l=2}^{k} f_{X_l}(X)$$  \hspace{1cm} (4.4)

Using the Lemma \[\square\] we then have the following bound on $P_e$

$$P_e < \sum_{l=2}^{k} g(X_l, X_1)$$  \hspace{1cm} (4.5)

This is the direct result of Lemma \[\square\] and \[\square\]. A looser bound is as follows which only depends on the two largest sets ($X_2$ and $X_1$).

$$P_e < (k - 1)g(X_2, X_1)$$  \hspace{1cm} (4.6)

This is because $f_{X_l}(X) < f_{X_2}(X), \forall l > 2$, meaning that starting with less nodes deciding on a specific state results in a lower probability of ending in that state.

Consider the case where $X_1^0 = \alpha_i N$, where $\alpha_1 > \ldots > \alpha_k$, and the result for the probability of error for two choices (3.5). Also, note that adding the undecided nodes does not change the probability of error. In the case
of binary consensus $X_1^0 + X_2^0 = (\alpha_1 + \alpha_2)N$. Accordingly we will have the following theorem,

**Theorem 8.** The probability of error for the multivalued consensus on $k$ choices $P_e$ satisfies

$$
\frac{1}{N} \log_2 P_e \leq -(\alpha_1 + \alpha_2) \left[ 1 - H \left( \frac{\alpha_1}{\alpha_1 + \alpha_2} \right) \right] + \frac{\log_2 (k-1)}{N} \tag{4.7}
$$

As before, $H(y) = -y \log_2(y) - (1-y) \log_2(1-y)$ for $y \in [0, 1]$.

Equation (4.7) shows that as with the binary consensus the probability of error decreases exponentially for the case of multivalued consensus. Also, it can be seen that the rate of the decay depends on $\alpha_1$, the portion of the majority state and $\alpha_2$ the state which has the highest number of votes among the other states. Therefore, similar to the binary case, adding an extra state improves the probability of error significantly compared with the voter model where $P_e = 1 - \alpha_1$ regardless of the size of the network (for complete graphs).

### 4.1.3 Convergence Time

We now try to find the convergence time of the algorithm. Let $X_1^t, \ldots, X_k^t$ denote the number of nodes in states 1, 2, \ldots, $k$ and $U^t$ denote the number of nodes in state $e$ at time $t$. We then have the following,

$$
\sum_{i=1}^{k} X_i^t + U^t = N, \forall t
$$
4.1 Multivalued Consensus

Now if we define states $x^t_{i,N} = X^t_i/N, 1 \leq i \leq k$ and $u^t_N = U^t/N$. Similar to the binary consensus case, the Markov process is a density dependent Markov Jump Process, and by the Kurtz’s theorem in [34], $(x^t_{1,N}, \ldots, x^t_{k,N}, u^t_N)$ converges on any compact time interval to $(x^t_1, \ldots, x^t_k, u^t)$, given by the following series of differential equations:

$$\frac{dx^t_i}{dt} = x^t_i u^t - x^t_i \left( \sum_{j=1}^{i-1} x^t_j + \sum_{j=i+1}^{k} x^t_j \right)$$

We know that $u^t = 1 - \sum_{i=1}^{k} x^t_i$, as a result:

$$\frac{dx^t_i}{dt} = x^t_i \left( 1 - x^t_i - 2 \sum_{j=1}^{i-1} x^t_j - 2 \sum_{j=i+1}^{k} x^t_j \right)$$

(4.8)

**Theorem 9.** Considering (4.8), for $x^0_1 > \ldots > x^0_k$, the time $T$ to reach $(x^T_1, \ldots, x^T_k)$ so that $x^T_1 \sim 1 - 1/N; \sum_{i=2}^{k} x^T_i \sim 1/N$ is the following:

$$T \sim \log N$$

(4.9)

Theorem 9 can be derived by solving the differential equation (4.8). The proof is as follows.

Note that $X^0_1 > \ldots > X^0_k$ meaning that nodes who are initially in state 1 held the majority. For the sake of simplicity, consider $x_i = x^t_i$ and therefore in the rest of the proof each one of the $x_i$s is a function of time $t$.

1 An overview of Kurtz theorem and approximation of Markov jump processes is given in Appendix B.
4.1 Multivalued Consensus

The following equations are the immediate results of (4.8) for \(x_1, \ x_2, \) and \(x_3\) respectively,

\[
\begin{align*}
    dx_1 &= x_1 \left( 1 - x_1 - 2 \sum_{i=2}^{k} x_i \right) dt \\
    dx_2 &= x_2 \left( 1 - x_2 - 2x_1 - 2 \sum_{i=3}^{k} x_i \right) dt \\
    dx_3 &= x_3 \left( 1 - x_3 - 2x_1 - 2x_2 - 2 \sum_{i=4}^{k} x_i \right) dt
\end{align*}
\] (4.10) – (4.12)

Subtracting (4.12) from (4.11) yields the following:

\[
    d(x_2 - x_3) = (x_2 - x_3) \left( 1 - 2x_1 - x_2 - x_1 - 2 \sum_{i=4}^{k} x_i \right) dt
\]

(4.13)

Consequently,

\[
    d \log(x_2 - x_3) = \left( 1 - 2x_1 - x_2 - x_3 - 2 \sum_{i=4}^{k} x_i \right) dt
\]

(4.13)

We also can derive the following from (4.10),

\[
    d \log x_1 = \left( 1 - x_1 - 2 \sum_{i=2}^{k} x_i \right)
\]

(4.14)

Equations (4.13), (4.14) then yield the following:
4.1 Multivalued Consensus

\[ d \log (x_1 (x_2 - x_3)) = \\
\left( 2 - 3x_1 - 3x_2 - 3x_3 - 4 \sum_{i=4}^{k} x_i \right) dt \quad (4.15) \]

We can further expand (4.15) by multiplying terms \(x_4, \ldots, x_k\) and using (4.8) as follows,

\[ d \log \left( x_1 (x_2 - x_3) \prod_{i=4}^{k} x_i \right) = \\
\left( k - 1 - (2k - 3) \sum_{i=1}^{k} x_i \right) dt \quad (4.16) \]

On the other hand, (4.8) results in the following for the multiplication of \(x_i\)s,

\[ d \log \prod_{i=1}^{k} x_i = \left( k - (2k - 1) \sum_{i=1}^{k} x_i \right) dt \quad (4.17) \]

Define \(T\) as the time when the consensus process converges. If we compute the integral of (4.16) and (4.17) and consider the common factor \(\sum_{i=1}^{k} x_i\) in both equations we obtain (note that \(x_i = x_i^t\) is a function of \(t\)):

\[ \int_{0}^{T} \sum_{i=1}^{k} x_i dt = \frac{1}{2k - 3} \left[ (k - 1) t - \log \left( x_1 (x_2 - x_3) \prod_{i=4}^{k} x_i \right) \right]_{0}^{T} \quad (4.18) \]
\[ \int_{0}^{T} \sum_{i=1}^{k} x_i dt = \frac{1}{2k - 1} \left[ kt - \log \prod_{i=1}^{k} x_i \right]_{0}^{T} \quad (4.19) \]
4.1 Multivalued Consensus

Note that (4.18) and (4.19) are the result of (4.16) and (4.17) respectively and their LHS is the same. The following equation can then be derived by the equality of the RHS of (4.18) and (4.19).

\[
\frac{k - 1}{2k - 3} T - \frac{1}{2k - 3} \left[ \log \left( x_1 (x_2 - x_3) \prod_{i=4}^{k} x_i \right) \right]_0^T = \frac{k}{2k - 1} T - \frac{1}{2k - 1} \left[ \log \prod_{i=1}^{k} x_i \right]_0^T
\]

We know when the algorithm converges, \( x_1^T \sim 1 - 1/N \). Furthermore, if we denote the number of the nodes corresponding to different choices (at the time of convergence) with \( \beta_i \), then \( x_i^T \sim \beta_i/N, \forall i > 1 \), and \( \sum_{i=2}^{k} x_i^T = 1/N \).

Then,

\[
\frac{1}{(2k - 1)(2k - 3)} T = \\
\left[ \frac{1}{2k - 3} \log \left( x_1 (x_2 - x_3) \prod_{i=4}^{k} x_i \right) - \frac{1}{2k - 1} \log \prod_{i=1}^{k} x_i \right]_0^T = \\
\frac{1}{2k - 3} \log \left( \left( 1 - \frac{1}{N} \right) \left( \frac{\beta_2 - \beta_3}{N} \right) \prod_{i=4}^{k} \frac{\beta_i}{N} \right) - \frac{1}{2k - 1} \log \left( \left( 1 - \frac{1}{N} \right) \prod_{i=2}^{k} \frac{\beta_i}{N} \right) + C
\]

Where the constant \( C \) depends on the initial fraction of the votes (i.e.
4.2 Mean-field approximation for Erdős-Rényi graphs

Consider the dynamics in which each node in the graph contacts any other node in the set $V$ with probability $p'/(N - 1)$ at each time step (instead of $1/(N - 1)$). This is equivalent of an Erdös-Rényi graph structure in which each pair of nodes is connected with probability $p'$. However, the structure
of the network in this setting is not fixed i.e. at each time step a new Erdős-
Rényi graph is constructed.

If we consider this as a mean field approximation of the dynamics in an
Erdős-Rényi graph the continuous time Markov process will be the same as
the complete graph. This is because in such a graph the number of edges is
$p'N$ and therefore the rate at which a node contacts its neighbours will be
the same as the complete graph \( \frac{N}{p'N} \times \frac{p'}{N-1} \approx 1/(N-1) \).

Further analysis is needed to prove this analytically for the case of regular
graphs.

### 4.3 Simulations

We now present the result obtained by simulations implemented in Matlab
\[55\]. The simulations in Figures 4.1, 4.2, and 4.3 show the probability of error
and convergence time for three different types of graphs namely complete,
Erdős-Rényi, and regular. The number of choices in these simulations is
three \((k=3)\) and accordingly there are three different groups of opinions at
the beginning of the consensus process with three different portions \(\alpha_1, \alpha_2\)
and \(\alpha_3\). For the case of error probability, networks have been simulated up
to the size of 2000 nodes. For the case of convergence time for complete
graphs networks have been simulated for sizes up to 15000 nodes while for
the case of regular and Erdős-Rényi they have been simulated for sizes up to
5000. For both of the probability of error and convergence time figures, the
consensus has been performed at each chosen size for few hundred times to
4.3 Simulations

give a good estimate (the bars show %90 confidence estimates accordingly). Furthermore, the degree of the regular graph is five and it has been generated using the configuration model (e.g. [16]). The parameter of the Erdös-Rényi graph (probability of having an edge between any two nodes) is 0.1.

Figure 4.1(a) shows that the probability of error of the algorithm for complete graphs decays exponentially along with our derived upper bound. Figure 4.1(b) shows that the convergence time grows logarithmically with $N$ which also confirms our result regarding the convergence time. Note that there is a constant difference between log $N$ and the convergence time ($\mathcal{O}(1)$ in (4.20)).

Figures 4.2(a), 4.2(b), 4.3(a), and 4.3(b) show the probability of error and convergence time of the algorithm for Erdös-Rényi and regular graphs respectively. These results are very similar to those of the complete graphs. The bounds in all these figures are the same, meaning that we used the same bounds for Erdös-Rényi and regular graphs. This was because although the structure of these graphs are not the same as the complete graphs, with some approximations it can be shown that the dynamics of the algorithm in Erdös-Rényi and regular graphs evolve with close proximity of those in the complete graphs. The derived results also confirm this, suggesting that the algorithm can be used in other types of graphs as well.
4.3 Simulations

(a) complete graph

(b) complete graph

Figure 4.1: Simulation results for $k = 3$ for Complete graphs. In the error plot (top) $\alpha_1 = 0.4, \alpha_2 = 0.35, \alpha_3 = 0.25$ and in the Convergence time plot (bottom) $\alpha_1 = 0.7, \alpha_2 = 0.2, \alpha_3 = 0.1$. The bars indicate %90 confidence estimates. Upper bound is given by (4.7).
Figure 4.2: Simulation results for $k = 3$ for Erdős-Rényi graphs. In the error plot (top) $\alpha_1 = 0.4, \alpha_2 = 0.35, \alpha_3 = 0.25$ and in the Convergence time plot (bottom) $\alpha_1 = 0.7, \alpha_2 = 0.2, \alpha_3 = 0.1$. The bars indicate %90 confidence estimates. Upper bound is given by (4.7).
4.3 Simulations

(a) regular graph

Figure 4.3: Simulation results for $k = 3$ for regular graphs. In the error plot (top) $\alpha_1 = 0.4, \alpha_2 = 0.35, \alpha_3 = 0.25$ and in the Convergence time plot (bottom) $\alpha_1 = 0.7, \alpha_2 = 0.2, \alpha_3 = 0.1$. The bars indicate %90 confidence estimates. Upper bound is given by (4.7).
4.4 Discussion

It is worth mentioning that using an extra state not only will not improve the probability of error in some cases such as path graphs, but it also will take longer than the voter model to converge to consensus.

Consider the algorithm on a path graph where voters for each choice are grouped together at the start of the algorithm, i.e. nodes number 1 to $\alpha_1 N$ are voting for choice 1, nodes $\alpha_1 N$ to $(\alpha_1 + \alpha_2) N$ vote for choice 2 and so on. In such a case the algorithm exactly develops as if we are using a voter model. The only difference is the fact that each node needs to contact its neighbour with a different opinion at least twice before accepting its opinion. For instance consider the following vector as a representative of three choices (choice 1: red, choice 2: blue and choice 3: green) on a path graph,

$$(r, r, r, r, b, b, b, g, g, g, g, g)$$

Here, the only change in the states of the nodes can happen at the nodes which are at the borders i.e. nodes 4 and 5 or nodes 7 and 8. If node 5 contacts node 4 the vector of states becomes,

$$(r, r, r, r, e, b, b, g, g, g, g, g)$$

where as before $e$ denotes the undecided note. The only way that node 5 can change its state to red is to contact node 4 again. Meanwhile, node 4 remains $r$ as it only has access to nodes with states $r$ and $e$. This is considering the
fastest way that node 5 can change its opinion. Clearly, this can take much longer.

In order to compare the voter model with consensus using an extra state analytically consider the case where $k = 2$. The above example changes to the following,

$$(r, r, r, r, b, b, b, b, b, b, b, b, b)$$

Note that binary consensus is a special case of multivalued consensus. It is then proved in [1] that the time of convergence of such scenario, $T_{rb}$ is as follows,

$$T_{rb} = 6RB,$$

where $R, B$ denote the number of nodes with colour $r$ and $b$ respectively. Furthermore the probability of ending in state $r$ is given by,

$$g(R, B) = R/B.$$ 

This means that the probability of error is the same as the voter model, while the convergence time is six times worse compared to the voter model.
Chapter 5

Optimization of binary interval consensus

Our main contribution in this chapter is to optimize the parameters of the binary interval consensus (Section 3.3) in order to speed up its convergence time. We first use empirical results to relate the convergence time of the binary interval consensus to the second largest eigenvalue of the rate matrix $Q$ that governs the consensus process. We will then optimize the eigenvalues of $Q$ utilizing the same techniques used for optimizing the convergence of the Markov chain of a graph in [28] (i.e. the FMMC problem). We recognize the fact that the optimization process needs to be done in a distributed fashion to be most useful in the context of distributed algorithms and therefore at the end of the chapter we suggest using the decentralization methods in [27] (also discussed in Section 3.5) in order to achieve this.

In Section 5.1 we give our empirical results for different types of graphs.
which relate the convergence time of the binary interval consensus algorithm to the second largest eigenvalue of the $Q$ matrix. In Section 5.2 we give our optimization method to reduce the convergence time. In Section 5.3 we provide the simulation results confirming the effectiveness of our optimization methods and finally in Section 5.4 we suggest a framework for decentralizing our optimization scheme.

5.1 Convergence time and eigenvalues of $Q$

In Section 3.3 we stated the bounds of the convergence time in terms of the parameter $\delta(G, \alpha)$ which depends on the portion of the majority $\alpha$ and the graph structure $G$. We also reviewed the optimization techniques for two well-known problems namely FMMC and averaging problem in Sections 3.4 and 3.5. In both problems, the optimization techniques are applied to the probability matrix that defines the possibility of nodes talking to each other. Concretely, in order to use the same techniques to optimize the convergence time of the binary interval consensus we need to relate the convergence time of the algorithm to the probability matrix $P$. We do this in two steps. We first relate the convergence time to the rate matrix $Q$. We then use (3.11) which relates the $Q$ matrix to $P$. This then facilitates the use of convex optimization techniques.

The very definition of $\delta(G, \alpha)$ by (3.17) makes its computation, let alone its optimization, computationally costly. To this end we relate delta to an alternative parameter for which the optimization procedure is more efficient.
5.1 Convergence time and eigenvalues of $Q$

We strongly believe as supported by empirical evidence that,

$$\delta \geq -(2\alpha - 1)\lambda_2(Q)$$  \hspace{1cm} (5.1)

We unfortunately have not been able to prove this identity, instead we have conducted extensive simulation for different graphs. The empirical evidence is in support of the conjecture that $\delta(G, \alpha)$ and $\lambda_2(Q)$ are of the same order. Figures 5.1(a), 5.1(b), 5.2(a), and 5.2(b) confirm the lower bound for Erdős-Rényi, regular, Power-law, and grid respectively (of size 16).

\textsuperscript{1}Our previous attempt along with its shortcomings can be found in Appendix C.
5.1 Convergence time and eigenvalues of $Q$

(a) Erdős-Rényi

(b) Regular

Figure 5.1: Simulation of $\delta(G, \alpha)$ and its bound (5.1) for Erdős-Rényi and regular graphs.
5.1 Convergence time and eigenvalues of $Q$

Figure 5.2: Simulation of $\delta(G, \alpha)$ and its bound (5.1) for Power-law and Grid graphs.

More specifically (5.1) and (3.20) result in the following relationship for the convergence time of the binary interval consensus,

$$\mathbb{E}(T) \leq \frac{2}{(2\alpha - 1)(-\lambda_2(Q))} (\log N + 1).$$  

(5.2)

Note that $\lambda_2(Q)$ is negative and hence the upper bound is positive. We can
5.1 Convergence time and eigenvalues of $Q$

find the convergence time empirically to test the upper bound given by (5.2).

Here the simulations can be done for graphs of larger sizes as the number of computations is much lower compared with computing $\delta(G, \alpha)$. Accordingly Figures 5.3(a) and 5.3(b) show the simulation results for Erdös-Rényi and regular graphs of size 400 while Figures 5.4(a) 5.4(b) show the simulation results for Power-law and Grid graphs of size 144.

![Erdos-Renyi graphs](image)

(a) Erdös-Rényi

![Regular graph](image)

(b) Regular

Figure 5.3: Simulation of the convergence time and its upper bound (5.2) for Erdös-Rényi and regular graphs.
5.1 Convergence time and eigenvalues of $Q$

Figure 5.4: Simulation of the convergence time and its upper bound (5.2) for Power-law and Grid graphs.

Using (5.2) as a proxy for the upper bound of $\mathbb{E}(T)$ (i.e. the convergence time) we can see that reducing $\lambda_2(Q)$ can result in the reduction of the convergence time and therefore we aim to optimize $\mathbb{E}(T)$ by optimizing $\lambda_2(Q)$. 
5.2 Optimizing binary interval consensus

In this section we show that the problem of the optimization of $\lambda_2(Q)$ is in fact a convex optimization problem and therefore it can be solved using SDP techniques. We now formulate our optimization problem. Equation (3.11) immediately results in the following:

$$\lambda_i(Q) = \lambda_i(P') - 1$$ (5.3)

This means that decreasing the eigenvalues of $P'$ will decrease the eigenvalues of $Q$ which changes the optimization problem of $Q$ to the optimization of $P'$. Note that $P'$ is both doubly stochastic and symmetric.

It is known that the sum of any number of the largest eigenvalues of a symmetric matrix is a convex function of the matrix (e.g. $\lambda_1(P') + \lambda_2(P')$ where $\lambda_1(P') \geq \lambda_2(P') \geq \ldots \geq \lambda_n(P')$) [28]. $P'$ is a stochastic matrix so $\lambda_1(P') = 1$ and consequently $\lambda_2(P') = (\lambda_1(P') + \lambda_2(P')) - 1$ is a convex function and can be optimized using convex optimization techniques such as SDP. There are other ways to prove the convexity of $\lambda_2(P')$ which we already discussed in Section 3.4.

The convex optimization problem will then be as follows:

$$\text{minimize} \quad \lambda_2(P')$$
subject to $P_{ij} \geq 0$, $P_{ij} = 0$ if $i, j \notin E$
and $\sum_j P_{ij} = 1, \forall i$ (5.4)
As it can be seen the optimization problem (5.4) is very similar to (3.21) and (3.45). More specifically both of the (5.4) and (3.21) optimization problems have a probability matrix in common. However, the FMMC problem focuses on reducing the maximum of $\lambda_2(\hat{P})$ and $-\lambda_n(\hat{P})$ while the optimization of the binary interval consensus needs to only reduce $\lambda_2(P')$ (recall that both $\hat{P}$ and $P'$ are symmetric and doubly stochastic). We therefore adjust the sub-gradient method in [27] to accommodate this difference.

5.2.1 Optimization

The optimization of $\lambda_2(P')$ is done through the elements of the $P$ matrix (i.e. $P_{ij}$s). In order to have an optimization vector (not a matrix) let $\omega$ represent the vector of the probabilities $P_{ij}$s. Concretely the vector $\omega$ will have $2m$ elements where $m$ is the number of edges.

We then need to assign a number $r$ for each edge $(i, j)$. This is denoted by $r \sim (i, j)$ ($i < j$, $r = 1, ..., m$). The elements of $\omega$ are then defined as follow:

$$\omega_r = P_{ij}, \omega_{-r} = P_{ji}$$

As the sum of the elements in each row of $P$ is 1 we do not need to optimize $P_{ii}$s. $P'$ can be then written as,

$$P' = I + \frac{1}{2} \sum_{r=1}^{m} (\omega_r B_r + \omega_{-r} B_{-r})$$
5.2 Optimizing binary interval consensus

where $B_r$ is a matrix of size $N$ associated with edge $r \sim (i,j)$ and defined as follows:

$$B_r = \begin{cases} 
B_{ri} = B_{rij} = -1 \\
B_{rij} = B_{rji} = 1 \\
0 & \text{elsewhere}
\end{cases}$$

(5.5)

To use the sub-gradient method, the convex optimization problem can then be defined as the following:

$$\text{minimize} \quad \lambda_2 \left( I + \frac{1}{2} \sum_{r=1}^{m} (\omega_r B_r + \omega_{-r} B_{-r}) \right)$$

subject to

$$1^T \Omega_i \Omega_i \Omega_i \leq 1, \forall i$$

and $\omega_r \geq 0$, $1 \leq |r| \leq m$

where $\Omega_i$ is defined as follows,

$$\Omega_i = [P_{ij}; (i,j) \in E]$$

Equivalently, $\Omega_i$ is the vector of non-zero elements in the $i$th row of the $P$ matrix (note that the sum of the elements in each row is 1).

Then, if $u$ is the eigenvector associated with $\lambda_2(P')$, the subgradient $f(\omega)$ will be,

$$f(\omega) = \frac{1}{2} \left( u^T B_{-r} u, ..., u^T B_r u \right)$$

and its components will be,

$$f_r(\omega) = \frac{1}{2} \left( u^T B_r u = -\frac{1}{2} (u_i - u_j)^2, r \sim (i,j), |r| = 1, ..., m \right)$$
Accordingly the steps in Algorithm 2 have been used for optimization at each step $k$, given a feasible $\omega$. Note that in Algorithm 2 the step size $\beta_k$ satisfies the diminishing rule,

$$\beta_k \geq 0, \beta_k \to 0, \sum_k \beta_k = \infty \quad (5.6)$$

This algorithm is very similar to the one used to solve the optimization problem (3.21) in Section 3.4. However, the sub-gradient $f(\omega)$ is built based on the eigenvector $u$ associated with $\lambda_2(P')$ while in Section 3.4 the sub-gradient was based on the eigenvector associated with $\lambda_2(\hat{P})$ or $\lambda_n(\hat{P})$ as the objective of the optimization was to minimize the $\zeta(\hat{P})$ (i.e. the maximum of $\lambda_2(\hat{P})$ and $-\lambda_n(\hat{P})$).

**Algorithm 2 Optimization**

1: $k \leftarrow 1$
2: repeat
3: // Subgradient Step
4: Calculate $f^{(k)}$ and update $\omega$, $\omega \leftarrow \omega - \beta_k f^{(k)}$
5: // Sequential Projection
6: $\omega_r \leftarrow \max \{\omega_r, 0\}, |r| = 1, \ldots, m$
7: for each node $i = 1, \ldots, N$, $L(i) = \{ r | \text{edge } r \text{ connected to } i \}$ do
8:   while $\sum_{r \in L(i)} \omega_r > 1$ do
9:      $L(i) \leftarrow \{ r | r \in L(i), \omega_r > 0 \}$
10:     $\gamma \leftarrow \min \{ \min_{r \in L(i)} \omega_r, \left( \sum_{r \in L(i)} \omega_r - 1 \right) / |L(i)| \}$
11:    $\omega_r \leftarrow \omega_r - \gamma, r \in L(i)$
12: end while
13: end for
14: $k \leftarrow k + 1$
5.3 Simulating the optimization

Here we present the results that we have found by simulating the optimization process for different types of graphs. Figures 5.5(a), 5.6(a), 5.6(b), 5.7(a) and 5.7(b) show the simulation results of the optimization for grid, Erdős-Rényi, Regular, and Power-law graphs (with different powers) respectively. In these simulations it is assumed that nodes are aware of their precise corresponding entry (and those of their neighbours) in the eigenvector associated with $\lambda_2(P')$.

![Simulation results for optimized and non-optimized versions of binary interval consensus for a grid of size 144. The bars indicate %90 confidence estimates.](image)

Figure 5.5: Simulation results for optimized and non-optimized versions of binary interval consensus for a grid of size 144. The bars indicate %90 confidence estimates.
5.3 Simulating the optimization

Figure 5.6: Simulation results for optimized and non-optimized versions of binary interval consensus. (Top) Expected time for an Erdös-Rényi graph of size 400. (Bottom) Expected time for a regular graph of size 324. The bars indicate (at least 90%) confidence estimates.
5.3 Simulating the optimization

(a) Power-law, \( h = 3 \)

(b) Power-law, \( h = 3.5 \)

Figure 5.7: Simulation results for optimized and non-optimized versions of binary interval consensus. (Top) Expected time for Power law graph of size 225 with \( h = 3 \). (Bottom) Expected time for Power law graph of size 225 with \( h = 3.5 \). The bars indicate \( 80 \) confidence estimates.

The \( x \)-axis in all the figures is the voting margin which is the difference between the initial portion of the nodes in the majority and the minority \((\alpha - (1 - \alpha) = 2\alpha - 1)\). Also, the initial votes in all these simulations have
been cast randomly to the nodes.

At each chosen $\alpha$ in the figures the consensus has been performed for tens of times to give a good estimate of the convergence time (the bars show confidence estimates). The degree of the regular graph is four and it has been generated by the configuration model in [16]. For the Erdős-Rényi graph the probability of having an edge between any pair of nodes is 5 percent. The Power-law graphs have been generated using preferential attachment (e.g. [24]) and modifying the MATLAB file in [56] where the portion of nodes with degree $i$ is proportional to $i^{-h}$.

As it can be seen the simulations confirm that the optimization algorithm can reduce the convergence time for different voting margins and different types of graphs. Also, the optimization works best when the voting margin is low (i.e. the initial difference between the number of votes is low). One explanation for this might be the following. When the voting margin is high i.e. when the difference between the majority and minority is high $P_{ij}$s become less relevant in defining the evolution of the states of the nodes. In this case the opinion of the neighbour is usually the one which is at the majority and therefore it does not make a difference which neighbour a node will choose to contact.

On the other hand, the effect of optimization is clearer for the case of Power-law graphs and it can be seen that the increase in the performance of the algorithm is much higher when the voting margin is low. Furthermore, the optimization performs better on the Power-law graphs with lower power $h$. 
One might relate this difference of the performance for different types of graphs to their specific degree distributions. The degree distribution in power-law graphs is not homogeneous (as opposed to Erdős-Rényi, grid, and regular graphs) and therefore different nodes have higher difference in their degrees. This means that the nodes with higher degrees are contacted more often by others (as they have more neighbours). In a setting where all the nodes have equal votes this results in delaying the consensus process as the diffusion of the opinions of the nodes with lower degrees happens at a lower rate.

We try to show the effect of the optimization on inhomogeneous networks with a simple example. Consider two simple star graphs connected as follows,
Figure 5.8: Two simple star graphs connected by one edge. Note that self-loops have not been shown in the figure.

Let $P$ and $P_{\text{before}}$ denote the probability matrix before and after optimization using our optimization algorithm the probability matrix changes as follows,
5.3 Simulating the optimization

(a) $P_{\text{before}}$

$$
\begin{bmatrix}
0.0730 & 0.1634 & 0.2073 & 0.3102 & 0.0652 & 0 & 0 & 0 & 0 \\
0.2694 & 0.7864 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.2877 & 0 & 0.7122 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.3101 & 0 & 0 & 0.8898 & 0 & 0 & 0 & 0 & 0 \\
0.0653 & 0 & 0 & 0.0361 & 0.1858 & 0.2126 & 0.2373 & 0.2616 & 0 \\
0 & 0 & 0 & 0.2890 & 0.8112 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.2128 & 0.7874 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.2374 & 0 & 0.7627 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.4618 & 0 & 0 & 0 & 0.7888 \\
\end{bmatrix}
$$

(b) $P$

$$
\begin{bmatrix}
0.0905 & 0.1860 & 0.1860 & 0.1559 & 0.4413 & 0 & 0 & 0 & 0 \\
0.1356 & 0.5142 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.1856 & 0 & 0.8142 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.1056 & 0 & 0 & 0.0192 & 0 & 0 & 0 & 0 & 0 \\
0.4420 & 0 & 0 & 0 & 0.0800 & 0.1994 & 0.1995 & 0.1995 & 0.1995 \\
0 & 0 & 0 & 0 & 0.1396 & 0.8604 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.1396 & 0 & 0 & 0.8604 & 0 & 0 \\
0 & 0 & 0 & 0.1396 & 0 & 0 & 0 & 0.8604 & 0 \\
\end{bmatrix}
$$

Figure 5.9: Simulation results for optimized and non-optimized versions of binary interval consensus. (Top) Probability matrix before the optimization and (Bottom) probability matrix after optimization.

It can be seen that the edge which connects nodes 1 and 5 can create a bottleneck for the process and not letting the votes flow easily in the network. In this case the optimization increases the probability of interaction between nodes 1 and 5 ($P_{15}$ and $P_{\text{before}15}$ in Figure 5.9). This results in the near 50 percent improvement of convergence time when the voting margin is narrow. Figure 5.10 shows the effect of the optimization on convergence time.
5.3 Simulating the optimization

Figure 5.10: Simulation results for optimized and non-optimized versions of binary interval consensus process in a graph of two connected stars.

Finally Figure 5.11 shows how Algorithm 2 can optimize $\lambda_2(Q)$ in an iterative manner. This is a sample optimization for an Erdős-Rényi graph of size 400 up to 500 iterations. The step size $\beta_k$ is chosen as $\frac{1}{\sqrt{k}}$ which satisfies (5.6).

Figure 5.11: Optimization of $\lambda_2(Q)$ for an Erdős-Rényi graph
5.4 Decentralization

An optimization scheme in the case of a distributed algorithm is most effective when it can be done in a distributed way. In this section we suggest using the decentralization techniques of the averaging problem in [27] which we discussed in Section 3.5. Consider the optimization process in the previous section. It can be seen that the subgradient $f$ can be computed using only local information if each node knows its corresponding entries in the eigenvector $u$ along with those of its neighbours. This allows the optimization to be done in a distributed way. For decentralization, we can use Algorithm [3] which is the Decentralized Orthogonal Iterations (DECENTRAL OI) algorithm in [53], also the same algorithm used in [27] to decentralize the subgradient method for averaging.

**Algorithm 3** Decentralization

1: Initialize the process with a random vector $y_0$
2: repeat
3: Set $y_k \leftarrow P' y_{k-1}$
4: $y_k \leftarrow y_k - \left( \sum_{i=1}^{N} \frac{1}{N} y_k \right) 1$ // Orthogonalize
5: $y_k \leftarrow y_k / ||y_k||$ // Scale to unit norm

The first step can be performed in a decentralized fashion as $P_{ij}$ is only non-zero when there is an edge between $i, j$. The second and third steps can also be done by using a distributed averaging algorithm such as [57]. Note that computing an approximation of the eigenvector results in computing an approximation of the sub-gradient. In [27] a result on the convergence of approximate sub-gradient methods (e.g. [58]) is used which proves that the sub-gradient method will converge even with considering the approxima-
tion error in finding the desired eigenvector in a distributed way. We are suggesting to use the exact same algorithm for decentralization. The only difference here is the use of matrix $P'$ instead of the $M$ matrix in [27], which governs the averaging process. The argument for proving the convergence of the sub-gradient method is also the same as [27].

As a final remark one might think of a scenario where the optimization process is done using a central entity before the nodes of the network start the consensus process. This way the optimal network can be set up in a centralized way where there will be no error in computing the eigenvector associated with $\lambda_2(P')$ and therefore no error in computing the sub-gradients. Such a scenario might be more appealing when the structure of the graph is fixed i.e. $P_{ij}$s will not change with time and therefore the optimization needs to be done only once. This is the case that we considered for our simulations in the previous section.
Chapter 6

Binary interval consensus in dynamic networks

In this chapter we present our framework for the distributed binary consensus in dynamic networks. Using the certain properties of interval consensus, in our proposed framework, new nodes can join the network at any point in time. There is no need for other nodes to refresh their state or change their way of communication and this can be achieved without the need of any extra memory.

However, nodes which want to leave the network will need one bit of memory for storing the initial value. They also need to run an extra procedure before leaving the network. We show that the time to run this procedure for nodes which leave the network does not depend on the size of the network for the case of complete graphs.
6.1 Binary interval consensus in dynamic networks

We already gave an overview of the binary interval consensus in Section 3.3 which is our base algorithm for dynamic consensus. We also reviewed the result regarding its convergence time in [7]. In Section 6.1 we present our algorithm which utilizes the binary interval consensus for dynamic networks. Although our algorithm is not restricted to a certain type of graph, we analyse its dynamics in the complete graphs using a direct analysis in the same section. We also simulate the result for complete graphs of different sizes.

6.1 Binary interval consensus in dynamic networks

By dynamic network we mean a network in which nodes can join or migrate from the network. For joining nodes we recommend using binary interval consensus without any changes and for the case of departing nodes we introduce an additional procedure which nodes should run before they can leave. This additional procedure can then guarantee the convergence to the right result.

6.1.1 Joining nodes

We claim the following lemma for a dynamic network where new nodes can join the consensus process,

**Lemma 10.** In binary interval consensus by new nodes joining the network at any given time step \( k \) the consensus will shift to the correct result based
on the majority at time $k$.

Proof. The proof follows by the fact that the binary interval consensus converges to the right result with almost sure probability. Of all the ways that the consensus can be reached amongst $N$ nodes, one way is that certain nodes will not communicate with any neighbours until some time step $k$. We denote the number of these specific nodes by $n_1$ and the number of others by $n_2$ (i.e. $N - n_1$). This dynamic is exactly the same as if the consensus starts in a network with size $n_2$ and then $n_1$ nodes join the network at time $k$.

In other words, using binary interval consensus at each time step the sum of all the nodes and consequently the average stays the same (by the conservation property (3.15)). By joining new nodes the sum of all the values of the nodes will be as if the new nodes had joined the network at the start of the consensus process and this guarantees the correct consensus at any time.

### 6.1.2 Departing nodes

Leaving the network however cannot be dealt with using the same approach. For example, if a node with initial value of 1 wants to leave the network when its current value is 0, the sum of the values of the nodes will not change. This means that the departure of the node with initial value 1 and a current value of zero, not only will not decrease the average but will increase it. Therefore, we need to set up a procedure for the nodes which need to leave the network to implement before their departure.

The key concept here is to make sure that the sum of the values of the nodes...
is the true representative of their initial values. Accordingly, if a node is leaving the network with a current value other than its initial value it should make the necessary changes to the sum of the values before its departure. For this it needs to first remember its initial value and consequently one bit of memory is needed for each node. We then propose the following procedure.

Consider node $i$ with the initial value 1 (respectively 0) which wants to leave the network. Its strategy will be as follows. Note that in the following rules, the update procedure are the same as before for other nodes. Only the node which is leaving should follow these rules,

- If its current state is 1 (respectively 0) it can leave the network right away.
- If its current state is either $0.5^-$ or $0.5^+$ it will set its state to 0 (respectively 1) and then wait to make contact with any node in state $0.5^-, 0.5^+, 1$ (respectively 0). It then leaves the network.
- If its current state is 0 (respectively 1). It will make contact with any node in state $0.5^-, 0.5^+$ or 1 (respectively 0) without updating its state, maintaining its state at 0 (respectively 1). It will then wait to contact any node in state $0.5^-, 0.5^+, 1$ (respectively 0). It then leaves.

The following lemma is then true.

**Lemma 11.** The above procedure will guarantee the correct consensus at any time after the departure of the nodes.

**Proof.** The proof follows by the fact that using the above procedure the sum of the values of all the nodes present in the network will be the sum of their
initial values and hence the consensus will remain correct at all times.

For instance, consider the following two vectors,

\[ x_i = (0, 0, 1, 1, 0, 0) \]

\[ x = (0.5^-, 0.5^-, 0, 0.5^-, 0, 0.5^-) \]

where \( x_i \) denotes the initial values of nodes at the start of the consensus process and \( x \) denotes their current value. As it can be seen the consensus process has already ended and all the nodes have chosen 0 as the majority state. Now if node 3 with initial value 1 wants to leave the network it has to reduce the sum of the values by 1. However, as the current state of node 3 is 0, it needs to contact one of the nodes 1, 2, 4, or 6 which are at state 0.5^−. Let us assume that it waits and finally contacts node 4. The vector of states becomes,

\[ x = (0.5^-, 0.5^-, 0, 0, 0, 0.5^-) \]

Note that node 3 has kept its state at 0. It now needs to contact any of the nodes 1, 2 or 6. If it contacts node 6, the following will be the vector of states,

\[ x = (0.5^-, 0.5^-, 0, 0, 0, 0) \]

And after node 3 leaves the network the vector of states becomes,

\[ x' = (0.5^-, 0.5^-, 0, 0, 0) \]

Therefore the sum of the values will be 1 as there is only one node present.
6.1 Binary interval consensus in dynamic networks

in the network with an initial value 1.

Using this framework for dynamic consensus there is no need to restart the process of reaching consensus when nodes join or leave. Furthermore, nodes that do not want to leave or join the network will not need to be aware of the dynamics. They will continue implementing the same algorithm that they had already been running. Only the nodes that want to leave the network have to implement an additional procedure before their departure.

6.1.3 The expected time it takes before a node can leave

It is clear from our proposed procedure that nodes leaving the network need to make contact with at most two nodes in other states before they can leave the network. Therefore it is useful to know how long this takes and whether it is feasible for the nodes to implement this extra procedure. We denote the time to implement the departure procedure by $T_{BD}$. To find $T_{BD}$ we have to consider several factors such as when the node is leaving (whether it is at the start, during, or after the consensus) and what its state is ($1$, $0.5^-$, $0.5^+$, or 0 which will then determine the number of contacts needed).

Here, we consider the case where a node leaves a complete network after the consensus has been reached. More specifically, consider a network where nodes decide to reach a consensus at specific time steps. Each consensus process $C_k$ starts at $t_{s_k}$ and finishes at $t_{f_k}$ and there is always a time gap between $C_k$s (i.e. $t_{s_{k+1}} > t_{f_k}$). In this scenario, nodes leave at intervals
6.1 Binary interval consensus in dynamic networks

$Y_k$s, where $Y_k = [t_{f_k}, t_{s_{k+1}})$. As before consider 0s as the majority. Also to consider the worst case assume that the node which is leaving is in state 0 with an initial value of 1 and therefore it needs at least two contacts with other nodes before its departure.

Following the stated conditions, as all the nodes are either in state 0 or 0.5, the departing node has to make contact with two 0.5-s before it can leave the network. The expected time for each node before it can leave the network is given by the following lemma.

**Lemma 12.** For a departing node in a complete graph using the dynamic binary consensus algorithm, the expected time to implement the departure procedure, $E(T_{BD})$ is,

$$E(T_{BD}) = \frac{N - 1}{2(1 - \alpha)N} + \frac{N - 1}{2(1 - \alpha)N - 1}$$

**Proof.** To find the expected time it takes before departure, let us first present the analysis in [7] for direct computation of the expected time of the depletion of nodes in state 1 in complete graphs (we denoted this by Phase 1 in Section 3.3).

Recall the notations we used in Section 3.3 and consider the time of convergence for complete graphs. If $\tau_i$ is considered as the time of the $i$th contact of a node in state 0 and 1 ($i = 1, ..., |S_1|$), it can be seen that the number of nodes in state 1 for any time $t \geq \tau_{|S_1|} = T_1$ is zero. Furthermore, if
6.1 Binary interval consensus in dynamic networks

\[ \tau_i \leq t < \tau_{i+1}, \quad |S_1(t)| = |S_1| - i \text{ and } |S_0(t)| = |S_0| - i. \]  

Also, at times \( \tau_i \),

\[ (|S_0| - i + 1, |S_1| - i + 1) \rightarrow (|S_0| - i, |S_1| - i) \]

It is then derived that if \( L_i = \tau_{i+1} - \tau_i \), \( L_i \) will be an exponential random variable with the following parameter,

\[ \gamma_i = \frac{(|S_0| - i)(|S_1| - i)}{N - 1} \quad (6.1) \]

where \( i = 0, \ldots, |S_1| - 1 \).

Using (6.1) and the fact that \( E(T_1) \) (i.e. the expected time of Phase 1) can be written as \( \sum_{i=0}^{(|S_1| - 1)} \gamma_i^{-1} \) leads to the following,

\[ E(T_1) = \frac{N - 1}{|S_0| - |S_1|} (H|S_1| + H_{|S_0|-|S_1|} - H_{|S_0|}) \quad (6.2) \]

where \( H_k = \sum_{i=1}^{k} \frac{1}{i} \). Accordingly,

\[ E(T_1) = \frac{1}{2\alpha - 1} \log(N) + 1. \]

Going back to the dynamic consensus framework and applying (6.2) will yield the expected time for the node in state 0 to make the first contact with a node in state 0.5−, \( E(T_{BD_1}) \) as follows:

\[ E(T_{BD_1}) = \frac{N - 1}{|S_{0.5-}| - 1} (H_1 + H_{|S_{0.5-}|-1} - H_{|S_{0.5-}|}) \quad (6.3) \]
where \( H_k = \sum_{i=1}^{k} \frac{1}{i} \).

Using the fact that after finishing the consensus \( S_{\{0,5\}} = 2(1 - \alpha)N \) and \( (6.3) \), \( \mathbb{E}(T_{BD_1}) \) will be given by:

\[
\mathbb{E}(T_{BD_1}) = \frac{N - 1}{2(1 - \alpha)N}
\]  \( (6.4) \)

Similarly, \( \mathbb{E}(T_{BD_2}) \) (the expected time for the departing node to make the second contact) will be given by:

\[
\mathbb{E}(T_{BD_2}) = \frac{N - 1}{2(1 - \alpha)N - 1}
\]  \( (6.5) \)

Note that after the first contact the number of 0.5-s will be reduced by 1. Finally \( \mathbb{E}(T_{BD}) \) will be given by:

\[
\mathbb{E}(T_{BD}) = \mathbb{E}(T_{BD_1}) + \mathbb{E}(T_{BD_2})
\]  

\[
= \frac{N - 1}{2(1 - \alpha)N} + \frac{N - 1}{2(1 - \alpha)N - 1}
\]  \( (6.6) \)

Equation \( (6.6) \) shows that when \( N \) is large the time that it takes for the node to leave the network (i.e. \( \frac{1}{1-\alpha} \)) will not grow with the size of the network.
Chapter 7

Conclusion and future work

7.1 Multivalued consensus

In Chapter 4 we showed that the consensus on multiple choices using an extra (undecided) state improves the performance for the complete graphs both in terms of the time of convergence and the probability of error (compared with the voter model). We proved this analytically and also confirmed it by running simulations for the complete graphs.

Furthermore, we showed that empirically using the algorithm is justified for the expander graphs such as Erdős-Rényi and regular. We also gave a mean-field approximation approach for the Erdős-Rényi graphs which gives the same bounds for its probability of error and convergence time.

We also stated that similar to the binary case using an undecided state for the consensus on $k$ choices does not give any advantage over the simple voter
model for graphs such as path.

Our results have been for complete graphs and although we gave a mean-field approximation for Erdős-Rényi graphs further analysis is needed to prove this with a direct analytical approach. This is also the case for the regular graphs.

7.2 Optimization of binary interval consensus

In Chapter 5 we presented an optimization scheme for the binary interval consensus process. We first related the bound of its convergence time to the second largest eigenvalue of the rate matrix governing the algorithm by conjecture. We also showed that the eigenvalues of the rate matrix $Q$ can be related to the eigenvalues of the $P'$ matrix which is a doubly stochastic matrix.

This then reduced the optimization problem to the optimization of the second largest eigenvalue of $P'$. We stated that utilizing the techniques used in FMMC and distributed averaging problem we can solve this optimization problem by changing it to a SDP problem and using the sub-gradient methods to solve it.

We confirmed the performance of our optimization process by simulating it for the Erdős-Rényi, grid, regular and Power-law graphs which are very different in nature. Our simulations showed that the optimization scheme works much better in graphs where the degree distribution of the nodes is not
uniform. It also enhances the performance more when the margin between the votes is narrow.

Finally we suggested the same decentralization techniques used in the averaging problem to decentralize our optimization method.

7.3 Binary interval consensus in dynamic networks

In Chapter 6 and based on the binary interval consensus algorithm we proposed a framework that dealt with the dynamics of a distributed binary consensus process in which nodes can join and leave the network. We showed that using this algorithm the consensus process can always converge to the majority of the present nodes even when the nodes join or leave the network. Note that the proposed algorithm is not restricted to the specific graph we have considered for the analysis.

7.4 Future work

In Chapter 4 we used a direct expansion of the binary consensus algorithm (using three states) to a multi-valued consensus scheme which proved to be useful in the case of complete graphs. However, there are other approaches in which multiple binary consensus processes can be used to solve multivalued consensus problems (e.g. [59] and [60]). We suggest starting by binary
interval consensus as the base algorithm. This can give other options for a multi-valued consensus process.

In Chapter 5 we derived an upper bound (in terms of $\lambda_2(Q)$) for the convergence time of the binary interval consensus using empirical results. We believe that it is possible to derive this bound analytically using a direct approach. To achieve this we suggest building on the approach of the authors in [7]. More specifically using their result on the eigenvalues of the $Q_S$ matrix which gives the upper bound of the convergence time in terms of the size of the network and $\delta(G, \alpha)$. Finding an upper bound which depends on $\lambda_2(Q)$ can prove the effectiveness of the optimization scheme for general connected graphs.

In Chapter 6 we started looking at dynamic networks by including the notion of leaving and joining nodes in the context of binary interval consensus. However, even though the initial results are promising our analysis is for the case of complete graphs and further work is required for other types of graphs starting by Erdős-Rényi and regular graphs. One might also want to include a rate of departure or arrival for binary interval consensus similar to the cases considered for the averaging algorithm in [12].

Throughout the thesis we mentioned different distributed algorithms using which nodes can reach a consensus on different number of choices. When it comes to real issues in practice, one might want to know which one of these algorithms is better for reaching consensus depending on different factors such as the graph structure, link failure probabilities, available resources, etc. Also, one might want to know if some of these algorithms can be combined
or even repeated to reach better performances. In [61], a framework for comparing different averaging algorithms has been introduced which includes the convergence time and cost (such as energy and message passing) of each algorithm. Motivated by their work one can come up with a framework for asynchronous consensus algorithms by starting from binary consensus algorithms. For instance, in [7], it is discussed that using three states for the binary consensus (Section 3.2) might be more beneficial if repeated (to reduce the error) instead of using the binary interval consensus for the case of complete graphs. This is because the former converges to an agreement in time $\log N$ (if it converges without error) while the latter converges in $\log N \over 2^\alpha - 1$. This is a simple example of a trade-off between multiple factors (i.e. error and speed in this case) and the decision should be made based on the application and available resources.
Bibliography


Appendices
Appendix A

First step analysis

As mentioned first step analysis is a technique which tries to find the absorption probabilities by defining a recursive formula based on the outcome of the first step of the process. It is easier to describe the first step analysis through an example. The Gambler’s ruin is one of the most famous probability problems in which first step analysis technique is used.

A.1 Gambler’s ruin

Starting with $k$ dollar, a gambler tries to earn an extra $n - k$ dollar to be able to buy a car worth $n$ dollar. He plays the following game. He tosses a fair coin and every time the coin shows head he wins 1 dollar. He loses one dollar otherwise. The possibility of him going bankrupt can then be given by the following solution.
Let $A$ define the outcome that the gambler goes bankrupt eventually and $B$ define the outcome of the first toss showing head. Also, let $P_k$ denote the probability of event $A$ happening starting with $k$ dollar. The following equations are then true,

$$P_k(A) = P_k(A|B)P + P(A|B^C)P(B^C)$$
$$= P_{k+1}(A)\frac{1}{2} + P_{k-1}(A)\frac{1}{2}.$$  \hfill (A.1)

This then results in the following,

$$P_k(A) = \frac{1}{2}(P_{k+1}(A) + P_{k-1}(A))$$
$$P_{k+1}(A) = 2P_k(A) - P_{k-1}(A),$$

and accordingly,

$$P_{k+1}(A) - P_k(A) = (2P_k(A) - P_{k-1}(A)) - P_k(A)$$
$$= P_k(A) - P_{k-1}(A)$$
$$= P_{k-1}(A) - P_{k-2}(A) = \ldots,$$

alternatively,

$$\forall k; P_k(A) - P_{k-1}(A) = P_1(A) - P_0(A).$$  \hfill (A.2)

We already know that $P_0(A) = 1$. Equation (A.2) then implies that for large $n$,

$$P_k(A) = 1 - \frac{k}{n}.$$  \hfill (A.3)
A.1 Gambler’s ruin

This means that eventually the gambler would go bankrupt by this strategy as $\mathbb{P}_k(A)$ approaches 1. This example shows how using first step analysis can help to solve the problem of finding the absorption probabilities. The same technique have been used in Section 3.2.
Appendix B

Density dependent jump

Markov processes

Density dependent family of Markov processes corresponds to a family of Markov processes with a parameter $N$ which can denote the population of or volume (e.g. in chemical processes). In this family states can be normalized and considered as densities. For instance, consider the example in [34], Chapter 8. In a chemical reaction molecule $A$ encounters molecule $B$ which results in the creation of molecule $C$. Furthermore the reverse is also true, meaning that molecule $C$ can decompose to $A$ and $B$ ($A + B \rightleftharpoons C$). Now consider the states of a Markov chain as the number of molecules i.e. $(X_A, X_B, X_C)$. The rates for the Markov chain that corresponds to such a
chemical process is taken as follows:

\[(X_A, X_B, X_C) \rightarrow (X_A - 1, X_B - 1, X_C + 1)\] with rate

\[\lambda_1 X_A X_B / N = N \lambda_1 (X_A/N)(X_B/N) \] (B.1)

and

\[(X_A, X_B, X_C) \rightarrow (X_A + 1, X_B + 1, X_C - 1)\] with rate

\[\lambda_2 X_C / N = N \lambda_2 (X_C/N) \] (B.2)

It is then shown in [34] that the transition rates of a density dependent family of Markov chains, \(X_N\) can be stated as,

\[q_{k,k+l}^N = N \beta_l \left( k \frac{1}{N} \right) , \] (B.3)

where \(\beta_l\) is a non-negative function. Let \(F(x) = \sum_l l \beta_l x\). The Kurtz theorem is then as follows.

**Theorem 13.** Suppose for all compact \(K \in \mathbb{R}^d\), \(\sum_l ||l|| \sup_{x \in K} \beta_l(x) < \infty\), and there exists a positive \(M\) such that

\[|F(x) - F(y)| \leq M|x - y| . \] (B.4)

Also, let \(\lim_{N \to \infty} X_N(0) = x_0 \) (\(X_N(t)\) denotes \(X_N\) at time \(t\)). Furthermore,

\(^1\text{For more details refer to [34].}\)
let $X(t)$ be a deterministic process which satisfies,

$$X(t) = x_0 + \int_0^t F(X(r))dr; \quad \text{(B.5)}$$

then

$$\lim_{N \to \infty} \sup_{r \leq t} |X_N(r) - X(r)| = 0 \ a.s. \ for \ all \ t > 0 \quad \text{(B.6)}$$

The implication of Kurtz theorem is that the limiting process $X_N(t)$ is in fact a deterministic process $X(t)$ given by a set of differential equations dictated by (B.5).

As we have seen in 4.1.3 the transition rates between the states $(x_{i,N}^1,\ldots,x_{k,N}^l;u_{i,N}^k)$ depends on densities and therefore could be considered as a density dependent family. We then use the Kurtz theorem to define differential equations which help us to find the convergence time of the consensus process.
Appendix C

Previous attempt on finding the lower bound of $\delta(G, \alpha)$

Here we present our previous attempt to find a lower and upper bound for $\delta(G, \alpha)$ and mention its shortcomings.

Our main result is the following:

**Theorem 14.** If $\delta(G, \alpha)$ is defined by Lemma 2 it would be bounded by the second largest eigenvalue of the rate matrix as follows,

$$-\lambda_2(Q)/4 \leq \delta(G, \alpha) \leq -\lambda_2(Q)$$

Theorem 14 yields that minimizing $\lambda_2(Q)$ would lead to the increase of $\delta(G, \alpha)$. This would then decrease the convergence time and hence the problem of optimization of the convergence time can be reduced to the problem
of optimizing the eigenvalues of the rate matrix $Q$. The proof of the theorem consists of the following lemmas.

**Lemma 15.** The following relationship exists between the diagonal elements of the $P$ matrix and $\delta(G, \alpha)$,

$$\frac{1 - \max P_{ii}}{2} \leq \delta(G, \alpha)$$

**Proof:** Consider the following result from [62] regarding the second real eigenvalue of nonnegative matrices. Define $A = [a_{ij}]$ as an irreducible nonnegative matrix. Furthermore, let $u$ and $v$ be the left and right eigenvectors of $A$ corresponding to $\rho(A) = \lambda_1(A)$ where $\lambda_1 > \cdots > \lambda_n$. Also, for $U \subset <n> = \{1, \ldots, n\}$ let $A(U)$ be the principle submatrix of $A$ whose rows and columns are in $U$ and set $\rho_m(A) = \max_{U \subset <n> : |U| = m} \rho(A(U))$. Also,

$$\epsilon(A, U) = \min_{\emptyset \neq V \subset <n> : |V| < U} \frac{\sum_{i \in V, j \notin <n> \setminus V} a_{ij} v_i u_j + a_{ji} v_j u_i}{\sum_{i \in V} 2(\rho(A) - a_{ii}) v_i u_i},$$

where $\emptyset \neq U \subset <n>$ and,

$$\epsilon_s(A) = \min_{\emptyset \neq V \subset <n> : |V| \leq s} \frac{\sum_{i \in V, j \notin <n> \setminus V} a_{ij} v_i u_j + a_{ji} v_j u_i}{\sum_{i \in V} 2(\rho(A) - a_{ii}) v_i u_i}$$

$$= \min_{U \subset <n> : |U| = s} \epsilon(A, U)$$
Then,

$$\rho_s(A) \leq \rho(A) - \frac{1}{2}(\rho(A) - \max_i a_{ii})\epsilon_s(A)^2$$  \hspace{1cm} (C.1)

Now using this result in our context for $P'$ matrix which is a doubly stochastic matrix we have the eigenvectors $u$ and $v$ to be unit vectors with elements equal to 1 which makes $\epsilon_s$ equal to 1. Also, $\rho(P') = 1$. Accordingly, (C.1) changes to the following for $P'$,

$$\rho_s(P') \leq 1 - \frac{1}{2}(1 - \max_i P'_{ii})$$

As $\lambda(Q) = \lambda(P') - 1$ and also considering the definition of $\delta(G, \alpha)$ in Lemma 2, this immediately results in the following,

$$\delta(G, \alpha) \geq \frac{1}{2}(1 - \max_i P_{ii})$$

where $P_{ii}$s are the diagonal elements of $P$ matrix. Note that $P'_{ii} = P_{ii}$.

**Lemma 16.** Consider the $\delta(G, \alpha)$ in Lemma 2 and $\lambda_2(Q)$, the second largest eigenvalue of the $Q$ matrix. The following relationship is then true,

$$\delta(G, \alpha) \leq -\lambda_2(Q)$$

*Proof:* Consider the $Q$ matrix as defined in the previous section. If $S = \{k\}$, (accordingly $|S| = 1, |S^C| = n - 1$) $Q_S$ would be equal to the $Q$ matrix except in the $k$th row in which all non-diagonal elements are zero and only the diagonal element $Q_S(k, k)$ is nonzero and equal to $q_k = -\sum_{i \in V} q_{i,k}$. If
we write the characteristic polynomial for row $k$, we will have the following,

$$\chi_{Q_S}(\lambda) = (q_k - \lambda)\chi_{Q_{S_k}}(\lambda)$$

where $Q_{S_k}$ is the matrix resulted by deleting the $k$th row and the $k$th column of $Q$ matrix. As $Q_S$ is the same as $Q_S^*$ (and therefore Hermitian), using Cauchy’s Interlace Theorem in [63], we will have the following result,

$$\lambda_2(Q) \leq \lambda_1(Q_S) \leq \lambda_1(Q)$$

Considering the definition of $\delta(G, \alpha)$ this would then prove the lemma. It can be seen that this also corresponds with the case where $\lambda_1(Q_S)$ is at its maximum as deleting other rows and columns would decrease the eigenvalues.

**Lemma 17.** Considering the $Q$ matrix which denotes the rate of the interactions between nodes, the following is then true.

$$-\lambda_2(Q) \leq 2(1 - \max P_{ii}) \quad (C.2)$$

**Proof:** We use the Gershgorin Theorem in [35]. The eigenvalues of $Q$ would then be bounded as follows,

$$|\lambda - q_{ii}| \leq \sum_{i \neq j} |q_{ij}| \quad (C.3)$$

Consequently $\lambda_2(Q)$ also obeys this bound and as $q_{ii} = -\sum_{i \neq j} q_{ij}$. Rear-
C.1 Problems with the analysis

ranging (C.3) would then yield

\[-2 \sum_{i \neq j} q_{ij} \leq \lambda_2(Q) \leq 0\]

Recall that \( Q = \frac{P + P^T}{2} \) and therefore, \( \sum_{j \neq i} q_{ij} = (1 - p_{ii})/2 \) which concludes the proof.

Lemmas 15, 16, and 17 would together prove Lemma 14. Using this result we now try to optimize \( \delta(G, \alpha) \) by optimizing \( \lambda_2(Q) \).

C.1 Problems with the analysis

In the proof of Lemma 15, \( \epsilon_s \) is not always 1 which means that \( \epsilon_s^2 \) cannot be taken out from (C.1). Furthermore, considering Gershgorin disks in Lemma 17, although usually most of the largest eigenvalues are concentrated at the disk with the centre of max \( P_{ii} \), technically max \( P_{ii} \) in equation (C.2) should change to min \( P_{ii} \).