Durham University

Doctoral Thesis

Randomised Algorithms on Networks

Author:
Sepehr Meshkinfamfard

Supervisor:
Dr. Tom Friedetzky

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

SCHOOL OF ENGINEERING AND COMPUTING SCIENCES

Wednesday 23rd March, 2016

Declaration of Authorship

Parts of the work presented in this thesis have been published in preliminary form [1] in collaboration with my colleagues. I declare that no part of the material presented in this thesis has previously been submitted for a degree.

DURHAM UNIVERSITY

Abstract

SCHOOL OF ENGINEERING AND COMPUTING SCIENCES

Doctor of Philosophy

Randomised Algorithms on Networks

by Sepehr Meshkinfamfard

Networks form an indispensable part of our lives. In particular, computer networks have ranked amongst the most influential networks in recent times. In such an ever-evolving and fast growing network, the primary concern is to understand and analyse different aspects of the network behaviour, such as the quality of service and efficient information propagation. It is also desirable to predict the behaviour of a large computer network if, for example, one of the computers is infected by a virus. In all of the aforementioned cases, we need protocols that are able to make local decisions and handle the dynamic changes in the network topology. Here, randomised algorithms are preferred because many deterministic algorithms often require a central control. In this thesis, we investigate three network-based randomised algorithms, threshold load balancing with weighted tasks, the Pull-Moran process and the coalescing-branching random walk. Each of these algorithms has extensive applicability within networks and computational complexity within computer science. In this thesis we investigate threshold-based load balancing protocols. We introduce a generalisation of protocols in [2, 3] to weighted tasks.

This thesis also analyses an evolutionary-based process called the death-birth update, defined here as the Pull-Moran process. We show that a class of strong universal amplifiers does not exist for the Pull-Moran process. We show that any class of selective amplifiers in the (standard) Moran process is a class of selective suppressors under the Pull-Moran process. We then introduce a class of selective amplifiers called punk graphs.

Finally, we improve the broadcasting time of the Cobra walk analysed in [4], for random regular graphs. Here, we look into the Cobra approach as a randomised rumour spreading protocol.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor Dr. Tom Friedetzky for his enthusiastic support throughout my Ph.D studies, and for his patience and knowledge. My sincere thanks also goes to my co-supervisor Dr. George Mertzios for his knowledge, support and immense motivation. I also would like to thank Dr. Petra Brenbrink for fruitful advice and help with collaborative work. Moreover, I wish to express my thanks to my examiners Dr. Stefan Danchev and Dr. Russell Martin.

My very special thanks goes to Dr. Sarah Dempsey for her priceless language advice and for proof-reading my thesis. Also I would like to thank Adam Poole for helpful programming advice.

I thoroughly enjoyed my time in the beautiful city of Durham. I wish to express my gratitude to the Durham University and the ACiD community. My sincere thanks goes to the Marie Curie International Training Network and the SCALUS project for giving me this fantastic opportunity and their financial support.

Last but not least, I would like to thank Dr. Sharry Borgan my mentor and friend for all his help and support.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements iii

Contents iv

List of Figures vi

List of Tables vii

1 Introduction 1
- 1.1 Introduction to Networks and Background to Thesis . . . 1
- 1.2 Randomised Algorithms . . . 2
- 1.3 Aims and Outline of the Thesis . . . 3
- 1.4 Definitions . . . 5
  - 1.4.1 Graphs . . . 5
  - 1.4.2 Markov Chain . . . 5
  - 1.4.3 Random Walks on Graphs . . . 6

2 Threshold Load Balancing with Weighted Tasks 9
- 2.1 Introduction . . . 9
  - 2.1.1 Balls into Bins Games . . . 10
  - 2.1.2 Resource-Based and User-Based Models . . . 10
  - 2.1.3 Related Work . . . 11
    - 2.1.3.1 Simple Balls into Bins games . . . 11
    - 2.1.3.2 Load Balancing Protocols with Migration . . . 12
    - 2.1.3.3 Threshold Based Protocols . . . 13
    - 2.1.3.4 Weighted Tasks . . . 14
  - 2.1.4 Our Contribution . . . 14
- 2.2 Notation . . . 15
- 2.3 Resource-Controlled Migration . . . 16
  - 2.3.1 Above Average Threshold . . . 17
  - 2.3.2 Tight Threshold . . . 20

2.4 User-Controlled Migration . . . 26
2.4.1 Model and Definitions . . . 26
2.4.2 Above Average Thresholds . . . 28
2.4.3 Tight Threshold . . . 32
2.5 Conclusion . . . 32

3 Evolutionary Dynamics of the Pull-Moran Process 33
3.1 Introduction . . . 33
3.1.1 Moran Process . . . 34
3.2 Related Studies . . . 35
3.2.1 Suppressors and Amplifiers of Selection . . . 37
3.2.2 The Pull-Moran (Death-Birth) Protocol . . . 37
3.2.3 Our Contribution . . . 38
3.3 Model and Definitions . . . 39
3.4 The Pull-Moran on Star . . . 40
3.5 Non-existence of Strong Universal Amplifiers . . . 43
3.6 The Bridge Thermal Theorem and Strong Suppressors . . . 46
3.6.1 A Class of Strong Selective Suppressor . . . 46
3.6.2 The Bridge Thermal Theorem . . . 48
3.7 The Pull-Moran on Regular graphs . . . 51
3.7.1 The Pull-Moran on Clique Graphs . . . 52
3.7.2 The Bridge Isothermal Theorem . . . 54
3.8 A Class of Selective Amplifiers for the Pull-Moran Process . . . 56
3.9 Conclusion . . . 60

4 Coalescing-Branching Walk 61
4.1 Introduction . . . 61
4.1.1 Random Walks . . . 61
4.1.2 Parallel Random Walks . . . 62
4.1.3 Rumour Spreading . . . 63
4.1.4 Cobra Walks . . . 65
4.2 Our Contribution . . . 66
4.3 Model and Definitions . . . 66
4.3.1 General Bounds and Matthews' Theorem . . . 67
4.3.2 Expander Graphs and Measures of Expansion . . . 68
4.3.3 Random Regular Graphs . . . 69
4.4 Cobra Push and Cobra Pull Protocols . . . 69
4.5 Cobra Push & Pull Protocol . . . 70
4.6 Simulations . . . 76
4.7 Conclusion . . . 76

Bibliography 78

List of Figures

3.1 PULL-Moran chain for star 41
3.2 PULL-Moran transitions in star 41
3.3 Manipulated PULL-Moran chain in star 43
3.4 The urchin graph with $2n$ vertices 47
3.5 The punk graph with $n \cdot k + n$ vertices $(k = 3, n = 4)$ 56
3.6 Fixation rate of the PULL-Moran process given an infected blade 57
3.7 Fixation rate of $f_r(G)$, f_2 and $\alpha(k, r) \cdot f_2$ with $n = k$ and $r = 10$ 59
3.8 Fixation rate of $f_r(G)$, f_2 and $\alpha(k, r) \cdot f_2$ with $n = k$ and $r = 5$ 59
3.9 Fixation rate of $f_r(G)$, f_2 and $\alpha(k, r) \cdot f_2$ with $n = k$ and $r = 3$ 60

4.1 Percentage of active/covered nodes for $k = 2, 6$ as a function of time . . . 77

List of Tables

1.1 Summary of hitting time and cover time of typical graphs [67] 8

4.1 Result summary of parallel random walks in [67] ($\forall \varepsilon > 0$) 62

Dedicated to my parents, Parivash Amin and Hassan Meshkinfamfard

In memory of my best friend Neal Smith Brodsky

Chapter 1

Introduction

1.1 Introduction to Networks and Background to Thesis

Networks are an indispensable part of our lives. We, and our relationships with each other form a large network called human society. Generally speaking, a *network* is a collection of individuals (also known as nodes) that are connected to each other through a certain behaviour. For example, biological networks are defined by relationships among biological organisms and economic networks capture the economic relationships among buyers (or consumers) and sellers (or producers) [5]. More recently, an emerging network exists in the form of computer networks, and these are amongst the most influential networks in our daily lives. According to the United Nations International Telecommunications Union (ITU), the number of Internet users reached 3 billion by the end of 2014. More strikingly, the number of global mobile-broadband subscriptions reached 2.3 billion by 2014, and this remains the fastest growing market segment, with continuous double digit growth rates in 2014 [6]. In such an ever-evolving and fast growing network, the primary concern is to understand and analyse different aspects of the network behaviour. One of these aspects is the quality of service. For example users of an Internet service provider should be able to access the Internet without any disconnection issues. Considering the scale of such a large network, in order to maintain a good service it is practically impossible to control the whole system centrally. Instead, it is of great importance to design protocols that can tackle this issue locally. Another fundamental aspect is to observe information dissemination throughout the network. There have been several *deterministic* and *randomised* protocols introduced and analysed in this respect [7–11]. On the one hand, it is important to facilitate efficient information propagation in applications such as broadcasting and load balancing. On the other hand it is desirable to predict the behaviour of a large computer network if, for example, one of the

computers (or servers) is infected by a virus. In all of the aforementioned cases, we need protocols that are able to make local decisions and handle the dynamic changes in the network topology. Here, randomised algorithms are preferred because many deterministic algorithms often require a central control, and they also struggle to cope with changes in the topology of a network.

1.2 Randomised Algorithms

Randomness plays an important role in a wide range of sciences including economy, biology and genetics, as well as in computer science where it is used for modelling and analysing algorithms. A *randomised algorithm* is a type of algorithm with a degree of randomness in making decisions. Intuitively, it can be considered as a protocol that flips a coin before determining what to do next. When designing *deterministic algorithms* the goal is to show that the algorithm resolves the problem correctly and as quickly as possible.

Deterministic algorithms always solve the problem correctly. The challenge is to find robust algorithms in which the number of steps taken to solve a problem is polynomial in the size of input. This is not always the case for deterministic algorithms. For many problems, a randomised algorithm is simpler and/or faster than a deterministic algorithm [12]. However randomised algorithms, as well as being dependent on the input size, also introduce an aspect of probability throughout their execution. This means that the outcome of a randomised algorithm may vary even for a fixed input. In addition to input data, randomised algorithms receive some probabilistic bits throughout their execution. Here, the goal is to design algorithms with a relatively good performance (*i.e.* short run-time), for every input.

There are two types of randomised algorithms; *Las Vegas* algorithms and *Monte Carlo* algorithms, originally named by László Babai [13]. A Las Vegas algorithm is a randomised algorithm that always gives the correct answer, as for a deterministic algorithm. However, the running time of a Las Vegas algorithm is a random variable. Conversely, a Monte Carlo algorithm is a randomised algorithm where the running time is deterministic but it gives a result that can be incorrect with a certain probability (*i.e.* correctness is not guaranteed). However, if a Monte Carlo algorithm is run for an arbitrarily large number of iterations, with independent random choices at each iteration, then the error probability can be made arbitrarily small. Randomised algorithms developed by Solo and Strassen [14] and Rabin [15] are some of the key studies on the application of such algorithms to number theory and computational geometry. Today, randomisation has become a very significant aspect in designing algorithms. There are two types of features

that make randomised algorithms more advantageous [16]; for the same problem, the running time or the required space of a randomised algorithm is often smaller than that of the best known deterministic algorithm. Also if we look in detail at the randomised algorithms designed to date, then we observe that they are very simple to understand and easy to implement. However, in analysing randomised algorithms we are faced with the same difficulties as for deterministic algorithms. The first goal is to design a reasonably efficient randomised algorithm that gives us a correct answer during a bounded running time. The second goal is to analyse the performance of the algorithm. In this respect the analysis, although difficult, can benefit from probabilistic methods due to the randomisation. Furthermore, when analysing the performance of distributed systems where the components form a network, randomised algorithms often require information from only the adjacent components in the network rather than global information about all components in the system. Therefore, compared to deterministic algorithms, randomised algorithms may require less space or storage.

When analysing randomised algorithms we are usually interested in the expected running time for the worst case scenario. This is the average number of time-steps, over all possible outcomes of the random bits, needed for the algorithm to perform when the worst input data is given.

Here our goal is to investigate three network-based randomised algorithms, threshold load balancing with weighted tasks, the Pull-Moran process and the coalescing-branching random walk. These goals are explained in the following section.

1.3 Aims and Outline of the Thesis

In this thesis, we investigate three network-based randomised algorithms, threshold load balancing with weighted tasks, the Pull-Moran process and the coalescing-branching random walk. Each of these algorithms has extensive applicability within networks and computational complexity within computer science.

In Chapter 2, we investigate *balls-into-bins* games in networks. The Balls-into-bins game is the process of randomly allocating m balls between n bins ($m > n$) under different conditions. Here, balls represent data or tasks and bins represent processors or storage. In particular we will investigate a specific type of this protocol where each bin is a node in a network and has a threshold that represents the capacity of the bin. Balls are initially allocated randomly between the bins and they are able to migrate from one bin into an adjacent bin in the network. Here, the goal is to find the estimated run-time of the process until all thresholds are satisfied. These algorithms provide a useful tool

for maintaining the quality of service for users in a network based system. Most of the conducted studies are based on uniform tasks (balls) in this respect. Here, our focus is on the generalisation of threshold-based models, which are introduced in [2, 3], to weighted tasks.

In Chapter 3, we study the dynamics of two different species, called *mutant* (or *infected*) and *non-mutant* (or *non-infected*), in different networks. In practice, mutants may represent influential people in a society (or a social network) or alternatively they can be seen as computers in a network that are infected by a virus. Our main focus is on the Moran process. In the Moran process [17], a network may consist of a non-mutated population with a single mutant individual. From this initial state, we are interested in the situation where the mutated individual infects other individuals and eventually takes over the entire population. The probability of this occurring is known as the *fixation probability*. Motivated by the study in [18, 19], here we investigate the death-birth update (or the Pull-*Moran* process as we call it) which investigates the likelihood that an individual is influenced by another in the network. The Pull-Moran protocol is independent of any global knowledge such as the total number of mutants in the network. This is advantageous in real-world applications because studying how different individuals are influenced by their neighbours (or circle of friends) is of great importance. The Pull-Moran also shows behaviour which are in contrast with the standard Moran process.

In Chapter 4, we study another network-based protocol called the *coalescing-branching* (Cobra) walk, introduced in [4], which is a modification of the simple random walk. The Cobra walk is designed to investigate the time taken for an epidemic disease to affect a large fraction, or all of the components, of a network. A Cobra walk is an iterative process where at each iteration every active node randomly chooses k neighbours in the network. These neighbours are labelled as active in the next iteration. In practice, any active node can represent an individual that has a piece of data or rumour or has been infected with a virus. Here, the goal is to estimate an upper bound on the *broadcasting* (or covering) time, that is the time taken for all nodes to have activated at least once. We are particularly interested in the broadcasting time of a Cobra walk on random regular graphs. Unlike [4], we analyse the Cobra walk within a broadcasting framework, where we improve the lower bound on the broadcasting time of random regular graphs.

Finally, we finish this chapter by introducing some definitions that are widely used throughout this thesis.

1.4 Definitions

Here, we will give an outline of the technical tools and methods used throughout the rest of this thesis. Most of the definitions in this section are mainly extracted from [38], [39] and [40].

1.4.1 Graphs

A *graph* is an ordered pair $G = (V, E)$ where V is the set of *vertices* (or *nodes*) and E is the set of *edges*. An edge is denoted by (u, v) where $u, v \in V$. The vertices u, v are *adjacent* if $(u, v) \in E$. For each $v \in V$, the *neighbourhood* $N(v)$ of v is the set of all vertices adjacent to v. Also let *degree* of v, denoted by $\deg v$ (or $d(v)$), be the total number of all vertices in $N(v)$. The *order* of a graph $n := |V|$ is the total number of vertices. The following are the definitions of the classes of graphs that are used in this chapter.

Definition 1.1. *A d-regular graph is a graph in which all vertices have the same degree d.*

Definition 1.2. *A graph G(V,E) is called complete (also known as clique) if G is a* $(n-1)$*-regular graph where* $|V| = n$.

1.4.2 Markov Chain

A *random process* $\boldsymbol{X} = \{X_t : t \in T\}$ is a collection of random variables defined on a set Ω. If t denotes time, which often is the case, $\boldsymbol{X}$ can be interpreted as a the value of a random variable changes over time. If Ω is a countably finite set then $\boldsymbol{X}$ is called a discrete *finite process*. Similarly, if T is a countably infinite set then $\boldsymbol{X}$ is called a *discrete time* process.

A (finite) *Markov chain* is a discrete time stochastic process of moving among the elements (or states) of a finite set Ω subject to a matrix $\mathbf{P}$ of transition probabilities. Given the current state $i \in \Omega$, the element $P_{i,j}$ of the *transition matrix* is the probability of moving to state $j \in \Omega$ from i at any time-step $t = 1, 2, \cdots$. Therefore, based on the definition $\mathbf{P}$ is *stochastic* which means $\sum_j P_{i,j} = 1$. More formally we have

Definition 1.3. *A sequence of discrete time random variables* $(X_t)_{t\geq 0}$ *is called a Markov chain if*

$$\mathbb{P}r\left[X_{t+1} = i_{t+1} | X_0 = i_0, X_1 = i_1, \ldots, X_t = i_t\right] = \mathbb{P}r\left[X_{t+1} = i_{t+1} | X_t = i_t\right] = P_{i_{t+1}, i_t}. \tag{1.1}$$

According to the above definition, the state X_{t+1} is independent of the history of the random process in getting to the state X_t, and only relies on X_t. Therefore, the aforementioned feature is called *memoryless property* or *Markovian property*.

Define $X_t = \boldsymbol{q^{(t)}} = (q_1^{(t)}, q_2^{(t)}, \ldots, q_n^{(t)})$ where $\boldsymbol{q^{(t)}}$ is the probability distribution vector at each time-step $t \geq 0$. It is easy to prove that $\boldsymbol{q^{(t+1)}} = \boldsymbol{q^{(t)}}\mathbf{P}$ where $\mathbf{P}$ is the transition matrix. Thus $\boldsymbol{q^{(t)}} = \boldsymbol{q^{(0)}}\mathbf{P}^t$ by induction, where $\boldsymbol{q^{(0)}}$ is the initial distribution.

Definition 1.4. *A distribution $\boldsymbol{\pi}$ on Ω is called the* stationary distribution *if $\boldsymbol{\pi} = \boldsymbol{\pi P}$, where $\boldsymbol{P}$ is the transition matrix of a Markov chain.*

Based on Definition 1.4, if there is a possibility for a Markov chain to reach a stationary distribution then it will keep that distribution in future.

1.4.3 Random Walks on Graphs

Definition 1.5. *Given a connected, undirected graph $G(V, E)$ with $|V| = n$ and $|E| = m$, a simple* random walk *starts at a node n_0. Being at a node n_t at time-step t, it moves to a node in $N(n_t)$ with probability $1/d(n_t)$. The sequence $(X_t = n_t)_{t\geq 0}$ is a Markov chain with transition matrix $\boldsymbol{P}$ where*

$$P_{i,j} = \begin{cases} 1/d(i) & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

In the following, we show that a random walk on a non-bipartite undirected graph $G(V, E)$ with $|V| = n$ and $|E| = m$ converges to a stationary distribution $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$, where

$$\pi(v) = \frac{d(v)}{2m}, v \in V \tag{1.3}$$

is the stationary distribution. Particularly the stationary distribution is the uniform distribution for all regular graphs, *i.e.* $\pi(v) = 1/n$ where $n = |V|$. Because $\sum_{v\in V} d(v) = 2m$ then we have

$$\sum_{v\in V} \frac{d(v)}{2m} = \sum_{v\in V} \pi_v = 1.$$

Therefore $\boldsymbol{\pi}$ is a probability distribution over all $v \in V$. Based on the definition of the stationary distribution, $\boldsymbol{\pi} = \boldsymbol{\pi P}$ where $\boldsymbol{P}$ is the transition matrix of the Markov chain

formed by a random walk. Given that $N(v)$ the neighbours of v, we have

$$\sum_{v \in N(v)} \frac{d(v)}{2m} \frac{1}{d(v)} = \frac{d(v)}{2m}.$$

Definition 1.6. *The hitting time $\mathcal{H}_{u,v}(G)$ of graph $G(V, E)$ is the expected time taken for a random walk to reach the vertex v when starting from u.*

Definition 1.7. *The maximum hitting time $\mathcal{H}(G)$ is*

$$\mathcal{H}(G) := max_{u,v} \mathcal{H}_{u,v}(G) \tag{1.4}$$

Definition 1.8. *For a non-bipartite graph G with transition matrix $\boldsymbol{P}$, if $dist(t) = \max_i |{P_{i,j}}^{(t)} - \pi(i)|$ then the mixing time $\tau(G) = \min\{t : dist(t) \leq \varepsilon\}$, where ${P_{i,j}}^{(t)} = (\boldsymbol{P}^t)_{i,j}$ and ε is an arbitrary constant.*

Intuitively, we are interested in the time $\tau(G)$ for which if $t > \tau(G)$ then the random walk distribution is almost $\boldsymbol{\pi}$.

If $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_n$ are the eigenvalues of the transition matrix of a random walk on a graph G and let $\lambda = min\{|\lambda_2|, |\lambda_n|\}$.

Theorem 1.9 (Theorem 5.1. [38])**.** *For a random walk starting at node i we have*

$$|{p_{i,j}}^{(t)} - \pi(i)| \leq \sqrt[n]{\frac{d(j)}{d(i)}} \lambda^t. \tag{1.5}$$

Definition 1.10. *Suppose $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$ be the eigenvalues of $\boldsymbol{P}$ the transition matrix of a random walk on a graph G. The* spectral gap *of $\boldsymbol{P}$ is defined as*

$$\mu := 1 - \max_{2 \leq i \leq n} \{|\lambda_i|\}.$$

Definition 1.11. *The cover time $\mathcal{C}(G)$ of a graph G is defined as the expected number of time-steps taken for a random walk to reach every node.*

Example 1.1. *The cover time on a path with n nodes is $\Theta(n^2)$.*

Consider a random walk on a path with nodes $1, 2, \cdots, n$. We are interested in finding the time it takes for a random walk to start from vertex 1 and end at vertex n. But first we need to observe $h_{i,j}$, $(i < j)$, the hitting time of reaching j having started the walk from i. For $2 \leq i \leq n - 1$ we have

$$h_{i,i+1} = \frac{1}{2} \cdot 1 + \frac{1}{2}(1 + h_{i-1,i} + h_{i,i+1}). \tag{1.6}$$

There are only two choices when starting from i; either walking to $i+1$, which takes 1 move, or walking to $i-1$ and then going back to $i+1$,which takes $1 + h_{i-1,i} + h_{i,i+1}$ moves. Solving Eqn. (1.6) for $h_{i,i+1}$, we get

$$h_{i,i+1} = 2 + h_{i-1,i}.$$

Solving the recurrence, subject to the fact that $h_{1,2} = 1$ yields

$$h_{i,i-1} = 2i - 1.$$

Hence

$$\begin{aligned} h_{1,n} &= \sum_{i=1}^{n-1} h_{i,i+1} = \sum_{i=1}^{n-1} (2i-1) \\ &= 2\sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} 1 \\ &= n(n-1) - (n-1) \\ &= (n-1)^2. \end{aligned} \tag{1.7}$$

In Table 1.1 you can find the hitting time and cover time of some of the typical graph classes.

TABLE 1.1: Summary of hitting time and cover time of typical graphs [67]

Graph	Cover time	hitting time
Cycle	$\Theta(n^2)$	$\Theta(n^2)$
2-dimensional grid	$\Theta(n \log^2 n)$	$\Theta(n \log n)$
d-dimensional grid, $d > 2$	$\Theta(n \log n)$	$\Theta(n)$
Hypercube	$\Theta(n \log n)$	$\Theta(n)$
Complete graph	$\Theta(n \log n)$	$\Theta(n)$
Regular Expanders	$\Theta(n \log n)$	$\Theta(n)$

Chapter 2

Threshold Load Balancing with Weighted Tasks

2.1 Introduction

In this chapter, our focus is on specific load balancing protocols for weighted tasks. Most of the balls-into-bins approaches, analysed from a theoretical point of view, were based on uniform balls. The novelty of our model lies in using weighted balls instead of uniform ones. Load balancing is the process of allocating tasks (or data) between servers (or hard disks) in an efficient way and/or within an efficient time. In computer science, we often face challenges regarding the allocation of data to a limited number of hard discs, each with a limited capacity for data storage. Similarly, processing tasks can be limited by a restricted number of servers. Tasks can be uniform or non-uniform and each server can have a different processing speed. Therefore, subject to different constraints we need different kinds of modelling strategies. Specifically, here we are interested in studying the protocols that allocate non-uniform (weighted) tasks between the servers with a defined threshold. In order to balance the load in such protocols we take advantage of *Balls-into-bins* schemes in the context of user-based or resource-based models. These concepts are explained in Section 2.1.2. In the following, after an introduction to the balls-into-bins games, we investigate two different protocols subject to the weighted tasks. In Section 2.1.3 we will review some of the related works which have been conducted in this framework and finally our main results will be shown in Sections 2.3 and 2.4.

2.1.1 Balls into Bins Games

Balls-into-bins games describe the process of randomly allocating m balls between n bins ($m > n$) under different conditions. Here balls represent data or tasks and bins represent processors or storage. The conditions are the rules under which the balls are distributed between the bins. In computer science balls-into-bins games are useful tools to analyse the load balancing. In other words, they are studied in order to build up new strategies that help us in both balancing the load and reducing the maximum load over all of the bins. Based on the conditions of various balls-into-bins protocols, we can have homogeneous or non-homogeneous (weighted) balls or bins, various distribution probabilities and even different networks of bins in the case of distributed systems. One of the basic models is the *single choice* balls-into-bins where each ball is allocated to a single bin selected *u.a.r*. We define the *load* of a bin as the number of balls allocated to that bin. Then, an interesting question is: what is the maximum load, over all bins, based on the number of balls m and bins n? Or, if the upcoming ball is allowed to select more than one random bin and is allocated to the least loaded one, does the maximum load change? In the model we develop in this chapter, we consider that the balls are already allocated into bins, which resemble vertices in a network. Also each bin is associated with a *threshold* that represents the capacity of that bin. The balls are allocated *u.a.r*, such that some bins may initially be filled above threshold. Balls can migrate from their current bin to an immediate neighbour, and these processes can occur in parallel. With each bin having a defined threshold, we are interested in analysing the time it takes to reach the state where the load in every bin is below the threshold. In the next section we discuss this protocol for both resource-based and user based models.

2.1.2 Resource-Based and User-Based Models

Managing *distributed systems* is a rising demand. In these systems, where the components (computers, servers etc.) are located on a network, global optimisation in order to maintain the quality of service may no longer be efficient or valid. For instance, consider a popular software upgrade which has been released, and all the users want to download sizeable files of the newest version. In another example, consider the users of a wireless network wanting to receive the best service. In both examples, users try to make connection with the least-loaded server. Therefore, in order to find an efficient approach that guarantees the quality of the service, we require load balancing protocols in which the (selfish) users are able to move (migrate) from one server to another within a distributed network, see [2, 20].

When studying threshold-based load balancing in distributed systems, here we will consider a network of bins each with an identical threshold. We are interested in analysing the performance of this model which mostly depends on the maximum load of each of the bins. Here, the performance is defined by the balancing time (run-time), *i.e.* the time it takes for the process to reach the state where the load in every bin is below threshold. We particularly focus on two different types of balancing protocols - the *resource-based* protocols and the *user-based* protocols. In resource-based protocols, each *overloaded* resource, that is the resource with a load above the threshold, is eligible to send tasks to its neighbouring resources. Moreover, our analysis of resource-based protocols will consider arbitrary graphs. For each graph, we have calculated two different balancing times subject to the tight threshold and the above average threshold. In user-based protocols, each task located at an overloaded resource decides weather or not to migrate to a neighbouring resource. From the network point of view, we only investigate the balancing time of user-based protocols applied on complete graphs. We review some of the studies that have been carried out on the related frameworks in the next section.

2.1.3 Related Work

The related approaches in this chapter are categorised based on four different types of load balancing models: the simple balls-into-bins games with no migration, load balancing protocols with migration, threshold based protocols and finally protocols that consider weighted tasks.

2.1.3.1 Simple Balls into Bins games

There are numerous papers that have discussed balls-into-bins games with uniform balls. In [21], Raab and Steger show that for $m = n$ in a single choice balls-into-bins game the maximum load is of the order of $\frac{\ln n}{\ln \ln n} \cdot (1+o(1))$. Later this model was generalised to the d choice model where each ball chooses $d \geq 2$ bins on his arrival and then allocates himself to the least loaded one. Particularly for the case where $d = 2$, (*i.e.* the two choice game) and $m = n$, further investigation shows that the maximum load is exponentially lower than that of a single choice game. In [22], Azar, Broder, Karlin and Upfal investigate the so called d-choice balls-into-bins game by introducing the GREEDY[d] protocol. They show:

Theorem 2.1 (Theorem 1.1. [22])**.** *Suppose that m balls are sequentially placed into n boxes. Each ball is placed in the least full box, at the time of the placement, among d boxes, $d \geq 2$, chosen independently and uniformly at random. Then after all the balls are placed,*

- *with high probability, as* $n \longrightarrow \infty$ *and* $m \geq n$*, the number of balls in the fullest box is* $(1 + o(1))\frac{\ln \ln n}{\ln d} + \Theta(m/n)$*;*
- *in particular, as* $n \longrightarrow \infty$ *and* $m = n$*, the number of balls in the fullest box is* $(1 + o(1))\frac{\ln \ln n}{\ln d} + \Theta(1)$*.*

Regarding the *weighted* balls-into-bins games, there has much research carried out in both single choice [23–25] and multiple choice [26] balls-into-bins games. We will discuss the weighted case in Section 2.1.3.4 in more detail.

2.1.3.2 Load Balancing Protocols with Migration

Various centralised and distributed load balancing protocols have been developed and analysed. In [20], Vöcking outlines *selfish* load balancing techniques including efficiency analysis and computational complexity. The goal is to reach the *Nash Equilibrium*, *i.e.* the state where no user can improve their quality of service via migration. In [27], Goldberg introduces *randomised local searches* where the tasks randomly make self-improving moves independent of each other in a distributed system. Initially, each task is allocated to a resource then, in every time step, a task and a resource are chosen *u.a.r* and the task moves to the resource if the resulting cost is lower. The proposed randomised algorithm shows an improvement in the expected time to reach a Nash equilibrium. Even-Dar and Mansour [28] analyse a load balancing protocol with three distinct features: (1) all identical tasks that share a resource will adopt the same policy, (2) no user moves from an under-loaded resource, and (3) no user can move from an overloaded resource to another overloaded one. Here an overloaded (under-loaded) resource had a load greater (less) than m/n where n and m are the number of resources and the number of tasks respectively. It is shown that the balance state (Nash equilibrium) is reached within $O(\log \log m + \log n)$ time steps for identical tasks. However, the number of time steps is polynomial for non-uniform tasks even for only two resources. The authors in [29] propose another method in which, unlike [28], each task only needs to know the load of its current resource and the load of a randomly-chosen alternative. They show that their approach converges to a Nash equilibrium in an expected time of $\log \log(m) + n^4$ steps. In [30], Brenbrink, Hoefer and Sauerwald, for identical machines in a network $G(V, E)$, find an expected time of $O(\Delta/\mu_2 \cdot (\ln m + \ln n) + |E| \cdot \Delta/\mu_2))$ to reach a Nash equilibrium, where Δ is the maximum degree in the network, μ_2 is the second smallest eigenvalue of the Laplace matrix of G and m, n are the number of tasks and machines respectively. They generalise the work in [28] to load balancing over machines (resources) with different speeds in complete graphs. Particularly, they show that their protocol converges to the Nash equilibrium

in $O(\ln(m) \cdot poly(n) \cdot poly(s_{max}))$ on any graph, where s_{max} is the maximum speed over all machines.

When analysing weighted tasks in a threshold based system, we make a distinction between threshold based protocols and the protocols considering weighted balls, and review the related works belong to each of these categories separately.

2.1.3.3 Threshold Based Protocols

In [31], the authors investigated a parallel balls-into-bins protocol where excess balls, subject to a given threshold T, are *rethrown* in the next round. Here, they analyse the trade-off between the threshold T and the number of rounds r. In particular they find for a threshold $T = \Omega(\sqrt[r]{\log n \log\log n})$, r rounds of communication are sufficient to balance the load, where n is the number of unit-sized balls as well as bins. From a threshold point of view, the work presented in [2, 3] is the most relevant to our study. The protocols discussed in [3] are user-controlled and resource-controlled and both models are applied on complete graphs. In case of above average threshold, for both user-controlled and resource controlled, they lower bound the balancing time by $\log(m)$, where m is the number of balls. Also, by applying an arbitrary threshold, they show a bound of $m \cdot \log^2(n)$, where n is the number of bins, for balancing time for the user-controlled case. The balancing times of both user-controlled and resource-controlled models demonstrated in [2] are basically the generalisation of the protocols in [3] to arbitrary graphs. For resource-controlled protocols, in [2], the authors show a bound of $O(\mathcal{H}(G) \cdot \log(m))$ for the balancing time, where $\mathcal{H}(G)$ is the maximum hitting time between any pairs of nodes in an arbitrary graph G. For *above average* threshold, *i.e.* the threshold of each resource is $> \frac{m}{n}$, they show the balancing time is roughly of the order of $O(\tau(G) \cdot \log(m) + \mathcal{H}(G) \cdot \log(n))$, where $\tau(G)$ is the mixing time of graph G, and the bound is tight. As for the user-controlled case, they introduce two different balancing stages. In the approximate balancing stage, similar bounds of $O(\mathcal{H}(G) \cdot \log(m))$ and $O(\tau(G) \cdot \log(m))$ are shown for arbitrary and above average threshold respectively. Moreover, they show that the time taken to reach a completely balanced state, is bounded by $O(n^5 \cdot \mathcal{H}(G) \cdot \log(m))$. In [32], the authors introduce two sequential balls-into-bins protocols called *Adaptive* and *Threshold*. In both approaches they show that, within $O(m)$ rounds of allocation time, it is possible to achieve a load of $\lceil \frac{m}{n} + 1 \rceil$ which is close to the optimal maximum load m/n.

2.1.3.4 Weighted Tasks

The idea of weighted balls was first introduced in [23], where the authors adapt the idea of *parallel* balls-into-bins games in [33] and generalise it to weighted tasks. They show a maximum load of $\gamma \cdot (m/n \cdot w_{avg} + w_{max})$ within $O\left(\log\log n/(\log\gamma \cdot ((m/n) \cdot \Delta + 1)\right)$ rounds of communication, where w_{max} and w_{avg} are the maximum and average weights respectively and $\Delta = w_{avg}/w_{max}$. *Single-choice* and *multiple-choice* balls-into-bins with weighted tasks are investigated in [34]. Here, the authors analyse the aforementioned protocols subject to different weight distributions as well as the order in which balls are allocated. In [35], the authors investigate the two-choices balls-into-bins procedure where m weighted tasks are allocated into n bins. Based on *the power of two choices*, each task is allocated to the least loaded of the two randomly chosen bins. Particularly, the authors show that as long as the weight distribution has a finite second moment and satisfies a mild technical condition, the gap between the load of the heaviest bin and the load of the average bin is independent of the number balls. In [36] the authors introduce the $(1+\beta)$-*choice* balls-into-bins where each ball is allocated to a random bin with probability $1-\beta$ and the least loaded of two randomly chosen bins with probability β, for some parameter $\beta \in (0,1)$. For the aforementioned process, the authors show that the gap between the load of the most loaded bin and the average is $\Theta(\log n/\beta)$, independent of m. Moreover, they show that the gap remains $\Theta(\log n/\beta)$ in the weighted case for a large class of weight distributions. A user-controlled balls-into-bins game is investigated in [29] where m balls are randomly distributed among n bins at the beginning. The approach is working in parallel rounds. In every round each ball is free (with a certain probability) to choose a bin randomly and allocate himself to that bin if the load of the selected bin is less than the load of the bin he currently resides in. The upper bound of the balancing time was determined to be $\log\log(m) + n^4$ for this case. The generalization of the user-controlled protocol in [29] to weighted balls is analysed in [37]. Specifically they find an upper bound of $O(nm\Delta^3\epsilon^{-2})$ for the expected time to reach an ϵ-Nash equilibrium. The latter protocols are then generalised to weighted balls and non-uniform resources with different *speeds*.

2.1.4 Our Contribution

Here, we investigate threshold based load balancing protocols for weighted tasks. We generalise the work developed in [2, 3] to the case of weighted tasks (users). Particularly, we study resource-controlled and user-controlled protocols. We analyse each class of the aforementioned protocols based on two types of thresholds. These are the loose (or above average) threshold, which is larger than the average load W/n by some constant

factor, and the tight threshold which is the average weight W/n. Here, W is total weight of all tasks and n is the number of resources. We will present our analysis for the two categories of protocols, with respect to above average and tight thresholds.

In Section 2.3 , we study resource-controlled protocols on an arbitrary graph G. For the above average threshold case in Theorem 2.6 we show that the balancing time is $O(\tau(G) \cdot \log m)$ with high probability, where $\tau(G)$ is the mixing time of a random walk on G and m is the number of tasks. Our bound, which is independent of the weights of the tasks, improves the expected balancing time $O(\mathcal{H}(G) \cdot \log n + \tau(G) \cdot \log m)$ for the uniform tasks presented in [2]. Here $\mathcal{H}(G)$ is the hitting time of a random walk on G. As for the tight threshold case, in Theorem 2.11, we show that the expected balancing time has a bound of $O(\mathcal{H}(G) \cdot \log m)$. This bound is tight (see Observation 2.12), and independent of weight and it matches the bound in [2] for uniform tasks.

In Section 2.4, we apply user-controlled protocols to complete graphs. For the above average threshold case, in Theorem 2.17, we show a bound of $O(w_{max}/w_{min} \cdot \log m)$ on the expected balancing time, where w_{max} and w_{min} are the maximum and minimum weight of all tasks. For tight thresholds, in Theorem 2.18 we show a bound of $O(w_{max}/w_{min} \cdot \frac{\log m}{n^2})$. Our bounds, for both above average and tight thresholds, match the abounds in [3] for unit-weight tasks. However, for weighted tasks, our bounds include an additional factor of w_{max}/w_{min}.

2.2 Notation

Assume $[m] = \{1, 2, \cdots, m\}$ are a set of tasks and $[n] = \{1, 2, \cdots, n\}$ are set of resources. Assume the resources are connected via an arbitrary graph denoted by $G(V, E)$. Let $d(i)$ be the degree of node $i \in [m]$ and Δ be the maximum degree, *i.e.* $\Delta = max_i d(i)$. Define $w_i \in \mathbb{N}$ to be the associated weight of each task $i \in [m]$. Define $w_{max} = max_i w_i$ be the maximum weight. Similarly let w_{min} be the minimum weight where $w_{min} \geq 1$. If this is not the case then we can simply adjust other parameters such that $w_{min} = 1$. Let $W = \sum_{i=1}^{m} w_i$ represent the total weight of all tasks. In our setting all of the actions are taking place in an iterative fashion. If $x_r(t)$ denotes the load of resource r at time t then we define

$$x(t) = (x_1(t), x_2(t), \cdots, x_n(t))$$

to be the load vector of the system at the beginning of time t, *i.e.* before any balancing takes place. Let

$$X(t) = (X_1(t), X_2(t), \cdots, X_n(t))$$

denote the state of the system at the beginning of time-step t, where $X_r(t)$ is the random variable that denotes the load of resource r at time t. Let $b_r(t)$ be the number of tasks on resource r at time t.

For an undirected, connected graph G let $P_{i,j}$ be the probability that the random walk moves from vertex i to vertex j. We consider standard random walks for non-regular graphs with transition matrix $\mathbf{P}$, where $P_{i,j} = 1/d$ for $i \neq j$ and $(i,j) \in E$ and $P_{i,i} = (d - d_i)/d$, where d is the maximum degree of the graph. We define $\mathbf{P^t}$ to be the tth power of $\mathbf{P}$. Then $P^t_{i,j}$ is the probability that a random walk starting from vertex i is located at vertex j after t steps. The stationary distribution of the random walk on G is called $\boldsymbol{\pi}(G)$ and it is the uniform distribution for this random walk. Note that the results in this chapter hold for all random walks in which the stationary distribution equals the uniform distribution.

2.3 Resource-Controlled Migration

In this section, we investigate the so called *resource-controlled migration* where resources (or nodes) are in charge of balancing the load in an iterative fashion. Tasks, currently assigned to a resource r in a distributed system, are only allowed to move to the neighbour(s) of r. As a consequence, every resource only needs to know about the global threshold and its own current load. Our main results in this section are based on two different sets of thresholds namely the *above average threshold* $\mathrm{Tr} = (1+\varepsilon)\cdot W/n + w_{max}$ where $\varepsilon \geq 0$, and the *tight* threshold $\mathrm{Tr} = W/n + 2w_{max}$. We need to tackle some new challenges for the weighted case which are not present in the uniform case. One of the main challenges is to have a well-defined value of the total weight of overloaded (unhappy) tasks on a resource, *i.e.* the sum of the weight of overloaded tasks on a resource r has to be unique. However, this may not be the case if we simply change the order of the non-uniform tasks in an overloaded resource. To resolve this issue we need to specify an ordering for tasks residing in a resource. Although the ordering we use is arbitrary due to the parallel nature of our protocol, having a fixed order is sufficient to find a solution. The other challenge relates to the definition of unhappy or overloaded tasks. Unlike the uniform-sized tasks case, in weighted case we may have the situation where a task with weight ≥ 2 is partially above and partially below the threshold. If such a task is not classified as unhappy then it may cause a potential decrease if it moves to an overloaded resource, in the case where the below threshold part is much larger than the above part. Hence, we need to take the partially above tasks into consideration.

We assume that every resource stores all its tasks in a stack data structure. However, in case of several tasks arriving at a bin simultaneously, within a time-step, the tasks are stored in an arbitrary order.

Definition 2.2. *The* height $h_r^i(t)$ *of task i on resource $r \in [n]$ at time t is the sum of the weights of all tasks in the stack data structure that are positioned before i.*

Definition 2.3. *Subject to the global threshold Tr, we define a task r to be partially above (cutting) the threshold Tr if $h_r^i(t) < Tr$ and $h_r^i(t) + w_i > Tr$.*

Based on Definition 2.3, each task stored at resource $r \in [n]$ can be *completely above*, *completely bellow* or *cutting* the threshold. Let $I_r^a(t)$ and $I_r^b(t)$ be the sets of tasks, on resource r at the beginning of time-step t, that are completely above and completely below the threshold respectively; they can also be empty. Define $I_r^c(t)$ to be the set containing the task on resource r that is *partially* above (cutting) the threshold at the beginning of time-step t.

Definition 2.4. *A task i is* accepted *by resource r if the height of the task i plus its weight is less than or equal to the threshold,* i.e. $h_r^i(t) + w_i \leq Tr$. *We call these tasks inactive. Similarly the tasks that are not accepted by a resource are called active.*

Initially tasks are randomly distributed between all of the resources. Then, according to Algorithm 2.3.1, the tasks in an overloaded resource that are labelled as inactive are allowed to move to a neighbouring resource that is randomly chosen. In this regard, we can think of each active task as undertaking a random walk through the network until it is accepted by a resource at which point it becomes inactive.

Note that when the task is labelled as inactive, *i.e.* when it is accepted by a resource at any time-step t, it will not leave that resource again.

Algorithm 2.3.1 Resource Controlled

for all resources r in parallel **do**
  if $x_r(t) >$ Tr **then**
    Remove any task $i \in I_r^a(t) \cup I_r^c(t)$ and reallocate the task to a neighboring resource that is chosen according to transition matrix P.
  Assign new heights to all migrated balls

2.3.1 Above Average Threshold

In this section, we assume that thresholds are larger by some constant factor than the average load W/n. More specifically, we assume that $\text{Tr} = (1+\varepsilon) \cdot W/n + w_{max}$ where ε is a non-negative constant. In the following we explain that the balancing time has the

order of $O(\tau(G) \cdot \log m)$. Intuitively, taking advantage of the above-average threshold, we find a constant fraction of the resources, called under-loaded resources, that are able to receive a task with any weight ($\leq w_{max}$) in the first step. Then we lower bound the probability of the active tasks landing on the under-loaded resources after running the algorithm for enough time-steps.

The next lemma gives us an estimate of the probability that a randomly chosen resource has a load less than or equal to the threshold. We use the outcome of the next lemma to prove the main result of this section. Here $X_r(t)$ is the random variable that denotes the load of resource r at time t.

Lemma 2.5. *If r is a resource chosen uniformly at random at time-step t. Then*

$$P[X_r(t) \leq Tr] \geq \varepsilon/(1+\varepsilon).$$

Proof. Base on a simple pigeon-hole argument, at any point in time (it could be either before or after task reallocations) there exists a fraction of $\varepsilon/(1+\varepsilon)$ resources that are able to store an additional task of any weight, $\leq w_{max}$, without violating the threshold. We prove it by contradiction. Assume that this is not the case which means that there are at least

$$(1-\varepsilon/(1+\varepsilon)) \cdot n + 1$$

many resources with a weight at least $(1+\varepsilon) \cdot W/n$. Hence

$$((1-\varepsilon/(1+\varepsilon)) \cdot n + 1)(1+\varepsilon) \cdot W/n > W,$$

Which is a contradiction. □

Theorem 2.6 is our main result with respect to the above average threshold and it holds for any arbitrary graph G. The result is stated in terms of the performance of random walks on G which forms a Markov chain. Here $\tau(G)$ is the mixing time, see Definition 1.8, of the aforementioned Markov chain. Particularly if G is a complete graph Theorem 2.6 achieves a bound of $O(\log m)$ for balancing time.

Theorem 2.6. *Let $Tr = (1+\varepsilon) \cdot W/n + w_{max}$. Assume that G is an arbitrary graph with mixing time $\tau(G)$. Let c be an arbitrary constant. Then, with a probability of $1 - n^{-c}$ all tasks are allocated after*

$$4(c+1) \cdot \tau(G) \cdot \frac{\log m}{\log(1+\varepsilon)}$$

time steps.

Proof. According to Algorithm 2.3.1, we can think of every active task performing a random walk subject to the transition matrix $\boldsymbol{P}$ until it is accepted by a resource, meaning that it reaches a resource that has a load small enough to be able to accept the task. Again based on the protocol, if a task is accepted by a resource then the task will become *inactive* and keep that status onwards. We divide the time-steps into *phases* of length $2\tau(G)$, where $\tau(G)$ is the mixing time of the Markov chain formed by the random walk on G. We fix a phase j and active task i and assume that task i remains active in the last step of phase j. Let $r_{i,j}$ denote the resource that the active task i visits at the last time-step of phase j. Here, we utilise a lemma introduced in [2].

Lemma 2.7 (Lemma 2.1. [2])**.** *Let G be an arbitrary graph. Let $\mathbf{P}$ be the transition matrix of a random walk on G. For any two nodes i, j and $t \geq 4\log n/\mu$, where μ is the spectral gap of $\mathbf{P}$,*

$$\pi_i - n^{-3} \leq \mathbf{P^t_{i,j}} \leq \pi_i + n^{-3}.$$

Therefore, assuming that $\tau(G) = 4\log n/\mu$, it follows that for every $k \in [n]$

$$\pi_i - n^{-3} \leq P[r_{i,j} = k] \leq \pi_i + n^{-3}. \tag{2.1}$$

Now from Lemma 2.5 together with 2.1, it follows that the probability that task i is accepted at the last time-step of phase j is at least

$$\frac{\varepsilon n}{1+\varepsilon} \cdot \left(\frac{1}{n} - \frac{1}{n^3}\right) \geq \frac{\varepsilon}{2(1+\varepsilon)}. \tag{2.2}$$

Let's assume that after ℓ many phases, with $\ell = (c+1)\cdot\log m/\log(1+\varepsilon)$, there still exists a task i that is still active. Being active after ℓ phases, a task i has not been accepted by any resource at any of the (last) time-steps of the ℓ phases. Therefore, according to Eqn. (2.2), the probability that the task i was not accepted by any resource at any of the (last) time-steps of each of ℓ phases, p_i, is

$$\begin{aligned} p_i &= \left(1 - \frac{\varepsilon}{2(1+\varepsilon)}\right)^{\ell} = \left(\frac{2+\varepsilon}{2(1+\varepsilon)}\right)^{\ell} \leq \left(\frac{1}{1+\varepsilon}\right)^{\ell} \\ &= (1+\varepsilon)^{-(c+1)\cdot\log m/\log(1+\varepsilon))} = \left(\frac{1}{m}\right)^{c+1}. \end{aligned} \tag{2.3}$$

Since the probability that all the tasks are allocated within ℓ step is

$$1 - \sum_{i=1}^{m} p_i,$$

the result occurs after $\tau(G) \cdot \ell$ many steps for $m > n$. □

2.3.2 Tight Threshold

In this section, we investigate a tighter threshold. We assume that

$$\mathrm{Tr} = W/n + 2w_{max}.$$

Our main result is Theorem 2.11, which is a bound for the balancing time which is stated in terms of the *maximum hitting time* $\mathcal{H}(G)$, see Definition 1.7. The following is the main idea of our proof.

Analysing the so-called resourced-control protocols under a tight threshold is more challenging than for the above average threshold setting where we could easily capitalise on a fraction of bins that are able to accept tasks of any weight. As in [2], here we need to introduce a potential function that counts the weight of *unhappy* (active) tasks, that are both above threshold and partially above threshold tasks. The next step is to show that the potential function does not increase, and this is obtained by following the definition of the potential. If we consider the weight of unhappy, *i.e.* tasks then in just one time-step they may end up being unhappy again, resulting in no change the potential, or they may be accepted by a resource that leads to a decrease in the potential. Finally we compare the resource-controlled protocols with another protocol, in which we assign each unhappy task to one of the resources such that the load of any resource is at most $W/n + w_{max}$, to show that after $2\mathcal{H}(G)$ time-steps the expected potential drop is large enough. Note that the potential drop is in expectation and therefore we need to use Theorem 2.10 (drift theorem) for bounding the balancing time.

We say that an allocation of the weighted tasks to the resources is *proper* if all the resources have a load below or equal to $W/n + w_{max}$. One can find a trivial proper assignment by using the simple *first fit* rule. As we mentioned, in order to analyse the resource-controlled protocol we introduce a potential function Φ that counts the total weight of the tasks that are either partially above the threshold, see Definition 2.3, or completely above the threshold. Let $I^a(t)$ be the set of all the tasks that are *above* threshold, *i.e.*

$$I^a(t) = \bigcup_{r\in[n]} I^a_r(t).$$

Similarly let $I^b(t)$ ($I^c(t)$) be the set of all the tasks that are *below* (*partially above (cutting)*) the threshold.

For any $t \geq 0$ the potential function Φ is defined as

$$\Phi(X(t)) = \sum_{i\in I^a(t)\cup I^c(t)} w_i. \tag{2.4}$$

For any $t > 0$ the potential change between subsequent states $X(t)$ and $X(t+1)$ is defined as

$$\Delta\Phi(t+1) = \Phi(X(t)) - \Phi(X(t+1)) \tag{2.5}$$

In the following observation we show that the Φ is non-increasing.

Observation 2.8. *For any $t > 0$ we have*

$$\Delta\Phi(t+1) \geq 0.$$

Proof. Based on the definition, at any time $t > 0$, any task $i \in [m]$ can either be in $I^c(t) \cup I^a(t)$ or $I^b(t)$ and not both of them. For the ease of presentation we assume that the protocol considers the task sequentially in an arbitrary order. In the following, we call a step where one of the tasks is considered a *sub-step*. We know that a task i is allowed to move to a randomly chosen neighbouring resource if and only if $i \in I^c(t) \cup I^a(t)$ at the beginning of step t. Suppose that task i is one of the tasks that are eligible to move and it moves to a randomly chosen neighbouring resource in sub-step t'. At the beginning of step $t+1$, there are only two possible cases for the task i, either $i \in I^c(t+1) \cup I^a(t+1)$ or $i \in I^b(t+1)$. For the first case the potential remains unaffected, and for the second case $\Delta\Phi(t+1) = w_i$. □

The next lemma finds a lower bound for the potential decrease during one time-step.

Lemma 2.9. *If $Tr = W/n + 2w_{max}$,*

$$\mathbb{E}\left[\Delta\Phi(t + 2\mathcal{H}(G)) \mid X(t) = x(t)\right] \geq \frac{\Phi(X(t))}{4}.$$

Proof. We consider a *phase* of length $2\mathcal{H}(G)$, where $\mathcal{H}(G)$ is the maximum hitting time, see Definition 1.7. At the beginning of each phase each active task is assigned to a resource, called the *target* resource, such that the maximum load of any resource is at most $W/n + w_{max}$. Then for every active task we place a token on its current resource and let the tokens perform a random walk of length $\mathcal{H}(G)$. If a token lands on the target resource of the corresponding task then the task is marked *blue*, otherwise the task is marked *red*. Let B and R be the set of blue and red tasks respectively. Also let $W(B)$ and $W(R)$ be the total weight of the blue and red tasks respectively. Note that

$$\Phi(X(t)) = W(B) + W(R)$$

since $\Phi(X(t))$ is the weight of the active tasks at the beginning of time-step t.

Let $X'(t)$ be the state in which all the task are assigned to the very same target resources as in $X(t)$. Furthermore, all the blue tasks are allocated to their targets. We define $\Delta\Phi'(t+2\mathcal{H}(G))$ to be the potential drop as a result of the blue tasks' allocation in $X'(t)$ in $2\mathcal{H}(G)$ time. We know that the expected time for a random walk on graph G to hit the target resource is the maximum hitting time $\mathcal{H}(G)$. We use the Markov inequality,

$$P[X > a] \leq \frac{\mathbb{E}[X]}{a}$$

where X is a random variable and $a > 0$. It shows that the probability of a random walk hitting its target resource after $2\mathcal{H}(G)$ time-steps is at least $1/2$. Therefore we have

$$\Delta\Phi'(t + 2\mathcal{H}(G)) \geq \Phi(X(t))/2.$$

Now we inspect the original process where every active task takes a random walk until it is accepted by a resource that has sufficient space to accept it. We want to show

$$\Delta\Phi(t + 2\mathcal{H}(G)) = \Phi(X(t)) - \Phi(X(t + 2\mathcal{H}(G))) \geq \frac{\Delta\Phi'(X(t + 2\mathcal{H}(G)))}{2} \geq \frac{\Phi(X(t))}{4}.$$

Without loss of generality, we can assume that each task, as long as it is not accepted by any resource, pursues its corresponding token. If it were accepted by one of the resources it is marked as inactive and its random walk terminates. We divide all the resources into two different sets. The resources that are not able to accept all of the incoming tasks throughout a phase, $2\mathcal{H}(G)$ time-steps, are labelled as FULL. Note that the load of a FULL resource has to be larger than $W/n + w_{max}$ by the end of a phase. A resource

that is not categorized as FULL is called *good*. A good resource is able to accept all of the incoming tasks within a phase without crossing the $W/n + w_{max}$ line.

Moreover, we categorise all blue and red tasks into three different sets. The set of blue tasks that are accepted by a good resource during a phase is denoted by B_g. Similarly, R_g are the red tasks that are accepted by a good resource. It follows that R_g and B_g are tasks which have not taken the position of any other task at its target resource. The tasks B_f (R_f) are blue (red) tasks that are accepted by a FULL resource in a phase. Lastly, the blue and red tasks that remain active and are not accepted by any resource at the end of a phase are denoted by B_n and R_n respectively. Therefore we have

$$\Delta\Phi'(X(t+2\mathcal{H}(G))) = W(B) = W(B_g) + W(B_f) + W(B_n)$$

and

$$\Phi(X(t)) - \Phi(X(t+2\mathcal{H}(G))) = W(B_g) + W(B_f) + W(R_g) + W(R_f).$$

Firstly, we show that $W(B_f) + W(R_f) \geq W(B_n)$ and then we prove our claim, *i.e.*

$$\Delta\Phi(t+2\mathcal{H}(G)) \geq \frac{1}{2}\Delta\Phi'(t+2\mathcal{H}(G)).$$

In order to show that $W(B_f) + W(R_f) \geq W(B_n)$, let ℓ_b be the total weight of the tasks that are accepted by resource b at the beginning of the phase. Then it follows that

$$W(B_n) \leq \sum_{b \in \text{FULL}} \frac{W}{n} + w_{max} - \ell_b. \tag{2.6}$$

The equality in Eqn. (2.6) is valid because $\sum_{b\in\text{FULL}} W/n + w_{max} - \ell_b$ is an upper bound for the total weight of the blue tasks that are able to be assigned to a FULL resource during a phase. Also we have

$$W(R_f) + W(B_f) \geq \sum_{b \in \text{FULL}} \frac{W}{n} + w_{max} - \ell_b. \tag{2.7}$$

Based on the definition of a FULL resource, which is a resource with a load larger than $W/n + w_{max}$, Eqn. (2.7) holds. Note that $\text{Tr} = W/n + 2w_{max}$.

Now we show that

$$\Delta\Phi(t+2\mathcal{H}(G)) \geq \frac{1}{2}\Delta\Phi'(t+2\mathcal{H}(G)).$$

Clearly $W(B_g) \geq 0$ and $W(R_f) \geq 0$, and therefore $W(B_g)+W(R_f) \geq 0$. Adding $W(R_f)$ on both sides gives

$$W(B_g) + 2W(R_f) \geq W(R_f).$$

Adding a $W(B_f)$ on both sides gives

$$W(B_g) + 2W(R_f) + W(B_f) \geq W(R_f) + W(B_f)$$

Since $W(B_f) + W(R_f) \geq W(B_n)$ we have

$$W(B_g) + 2W(R_f) + W(B_f) \geq W(B_n).$$

Dividing both sides by 2 gives

$$W(B_g)/2 + W(R_f) + W(B_f)/2 \geq W(B_n)/2.$$

Adding $W(B_g)/2 + W(B_f)/2$ on both sides gives

$$\begin{aligned} W(B_g) + W(R_f) + W(B_f) &\geq W(B_g)/2 + W(B_f)/2 + W(B_n)/2 \\ &= \frac{1}{2}(W(B_g) + W(B_f) + W(B_n)). \end{aligned}$$

The LHS is just $\Delta\Phi(t+2\mathcal{H}(G)) - W(R_g)$, and the RHS is $\frac{1}{2}\Delta\Phi'(t+2\mathcal{H}(G))$, so we obtain

$$\Delta\Phi(t+2\mathcal{H}(G)) - W(R_g) \geq \frac{1}{2}\Delta\Phi'(t+2\mathcal{H}(G)),$$

and as $W(R_g) \geq 0$ the claim of $\Delta\Phi(t+2\mathcal{H}(G)) \geq \frac{1}{2}\Delta\Phi'(t+2\mathcal{H}(G))$ follows. This concludes the proof of this lemma. □

We use the following *Drift Theorem* to prove the main result of this section. Note that there exists different versions of the so called drift theorem mainly to analyse the run-time of randomised algorithms and we have selected the following one.

Theorem 2.10 (Theroem 2. [41])**.** *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum s_{min}. Let $\{V(t)\}_{t\in\mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random variable that denotes the first point in time $t \in \mathbb{N}$ for which $V(t) = 0$. Suppose that there exists a constant $\delta > 0$ such that*

$$\mathbb{E}\left[V(t) - V(t+1) \mid V(t) = s\right] \geq \delta s \tag{2.8}$$

holds for all $s \in S$ *with* $\mathbb{P}[V(t) = s] > 0$. *Then for all* $s_0 \in S$ *with* $\mathbb{P}[V(0) = s_0] > 0$,

$$\mathbb{E}[T \mid V(0) = s_0] \leq \frac{1 + \ln(s_0/s_{min})}{\delta}. \tag{2.9}$$

Intuitively, Theorem 2.10 gives an upper bound on the expected run-time of a sequence of random variables, defined on a finite set, with a bounded decline in expectation.

Theorem 2.11. *Assume* $Tr = W/n + 2w_{max}$. *Let* $\mathcal{H}(G)$ *be the hitting time of the random walk on* G *with uniform stationary distribution. Let* T *be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}[T] = O\left(\mathcal{H}(G) \cdot \ln(W)\right).$$

Proof. As the potential change, within $2\mathcal{H}(G)$ time, is bounded by W from above, we can conclude the result using Theorem 2.10 (drift theorem).

□

The following Observation shows that the bound presented in Theorem 2.11 is tight for the protocol discussed in this section. Note that for weighted tasks the bound in Theorem 2.11 is not tight since the total weight W can be super-polynomial, *i.e.* it is not bounded above by any polynomial in m, the number of tasks. Therefore in Observation 2.12 we consider an amount m of unit-size tasks.

Observation 2.12. *There is a class of graphs such that the resource based protocol converges to a balanced state in an expected number of steps of* $\Omega\left(\mathcal{H}(G) \cdot \log m\right)$ *for tight thresholds.*

Proof. We use the same argument as [2] to show the tightness of the bound. In Theorem 3.7 [2], the authors investigate a graph $G'(V, E)$ that consists of two cliques V_1 and V_2 each of size $n/2$. The two cliques are connected by a total of k edges, where $1 \leq k \leq \frac{1}{5}n^2$. Every vertex in each clique is connected to at least $\lfloor k/(n/2) \rfloor$ and at most $\lceil k/(n/2) \rceil$ vertices of the other clique. Here instead of two cliques connected to each other by a total of k edges, we introduce the following graph G. Consider G consists of a clique K with $n-1$ nodes and a single node u. Moreover, we assume u is connected to exactly k nodes of the clique for some arbitrary $k < n$. First, we show that the maximum hitting time of G is $\Theta(n^2/k)$. Consider we are at any of the vertices in the clique, which are connected to u. Then we choose to go to u with probability k/n in total (k vertices connected to u). Therefore it takes about n/k steps for a random walk to choose to go to u. Now, if we are at one of the vertices in the clique, which is not adjacent to u, it

takes approximately $\Theta(n)$ to go to a vertex that is adjacent to u. Hence, in total, it takes $\mathcal{H}(G) = \Theta(n^2/k)$ steps to go from any vertex in the clique to u. As for allocating tasks, we first distribute the tasks on the nodes in K such that every node in K has a load of m/n and the remaining tasks are all allocated to an arbitrary node in K. We call this initial state x_0. To reach the balance state while starting from x_0, there are $\Phi(x_0) = m/n$ random walks starting from K that have to reach vertex u. We first need to bound the probability of these random walks staying in K within t time steps. We use the following lemma that concerns graph $G'(V, E)$ (defined above) in [2] to finish our proof.

Lemma 2.13 (Lemma 3.8. [2])**.** *Consider a random walk that starts at a vertex in V_1 with $\lfloor k/(n/2) \rfloor$ neighbours in V_2. Then, for any integer t, the probability that the random walk stays within V_1 for t step is at least*

$$4^{-\frac{16kt}{n^2}-\frac{1}{2}}$$

We can use the above lemma for our case by swapping V_2 with a vertex u with out loss of generality. Because all of the random walks are independent, after t steps at least one walk remains in the clique with a probability of at least $1-(1-4^{-\frac{16kt}{n^2}-\frac{1}{2}})^{\Phi(x_0)}$. For $t = \frac{\log_4 \Phi(x_0)-\frac{1}{2}}{16k} \cdot n^2$ the latter probability is $1-(1-\frac{1}{\Phi(x_0)})^{\Phi(x_0)} \geq 1-\frac{1}{e}$. Hence, we can achieve an expected balancing time of

$$\Omega\left(\mathcal{H}(G)\log(m/n)\right) = \Omega\left(\mathcal{H}(G)\log(m)\right)$$

for $m \gg n$. □

2.4 User-Controlled Migration

In this section, we will investigate the so called user-controlled protocols applied on complete graphs. According to user-controlled protocols every task located at an overloaded resource decides whether or not to migrate to a neighbouring resource.

2.4.1 Model and Definitions

As for user-controlled migration, we assume that the tasks are stored in the resources based on a stack data structure. Let G be an arbitrary graph that represents the network of a distributed system. Let n and m be the number of resources and the number of tasks respectively. Also W is the total weight of the tasks. Define w_{max} as the *maximum*

weight of any task. Recalling the Definitions 2.2 and 2.3 in Section 2.3, we assume that the *height* $h_r^i(t)$ of task i on resource $r \in [n]$ at time t is the sum of the weights of all tasks in the data structure that are positioned before i. Subject to the global threshold Tr, we define a task r to be partially above (cutting) the threshold Tr if $h_r^i(t) < \text{Tr}$ and $h_r^i(t) + w_i > \text{Tr}$. We say a resource is *over-loaded* if its load is larger than the threshold Tr. We are now ready to define the potential function.

Definition 2.14. *The potential of an over-loaded resource r at a time-step t is denoted by $\phi_r(t)$ and it counts the weight of the task which is partially above (cutting) the threshold (if there is any) together with the task(s) that is(are) above the threshold, if there is any.*

The potential $\Phi(t)$ at any time-step t is defined as follows

$$\Phi(t) = \sum_{i=1}^{n} \phi_i(t).$$

Let $x_r(t)$ be the load (total weight of the tasks in r) of resource $r \in [n]$ at time-step t. Also let $b_r(t)$ be the number of the tasks in r at time-step t. If it is clear from the context that t is fixed then we only use the notations ϕ_r, b_r, and x_r.

Then, according to the user-controlled migration protocols (Algorithm 2.4.1), tasks may leave an overloaded resource r with a probability $\alpha \cdot \left\lceil \frac{\phi_r(t)}{w_{max}} \right\rceil \cdot \frac{1}{b_r(t)}$, with $0 < \alpha < 1$, and move to a randomly chosen neighbouring resource.

Algorithm 2.4.1 User-Controlled

for all All users do in parallel **do**
  Let $r(i)$ be the resource storing user i.
  if $x_{r(i)} > \text{Tr}$ **then**
    i migrates to a resource chosen *u.a.r* with probability $\alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r}$.

In order to achieve an upper bound for the balancing time we follow almost the same line of argument presented in [2] with some modifications for the weighted case. According to the protocol, all the tasks located in an overloaded resource are allowed to leave their current resource with the same probability. This feature has both advantages and disadvantages. On one hand it facilitates the migration of the surplus tasks, which eventually leads to a decrease in potential. But on the other hand it motivates all the other tasks to leave and this acts to increase the potential. Therefore, we only need to show that the potential drop exceeds the potential increase.

Even though the potential is defined based on the weight of above and partially above threshold tasks, in reality our analysis and calculations are based on the *number* of tasks

above threshold, see Eqn. (2.10). We can explain the reason for using such a potential in an example. Consider a situation where there are some noticeable number of slightly overloaded resources with only a partially above threshold task of weight w_{max} and no above threshold tasks. In this specific case, it is likely that any of these large tasks will move to other slightly overloaded resources, since the migration probability is the same for every task in a resource, and this will increase the weighted based potential dramatically. As a consequence, we can address this issue by using a potential that counts the number of tasks located above the threshold.

2.4.2 Above Average Thresholds

In this section we assume that the threshold $\mathrm{Tr} = (1+\varepsilon)W/n + w_{max}$ where W is the total weight of the tasks and $\varepsilon > 0$. Analysing the potential change, we assume that the tasks are stored in a stack data structure in a resource. We presume that the tasks are leaving an overloaded resource one after another subject to the order of their heights in the stack. Furthermore, for the sake of analysis, we assume tasks are moving sequentially, *i.e.* tasks leave the first resource, and then the second resource and so on.

For a fixed t the potential $\Phi(t+1)$:

- decreases by w_i for every task i which was above the threshold $(h_r(t) + w_i > \mathrm{Tr})$ and then migrates to a resource r' such that $h_{r'}(t+1) + w_i \leq \mathrm{Tr}$.
- does not change for every task i which was above the threshold $(h_r(t) + w_i > \mathrm{Tr})$ and then migrates to a resource r' such that $h_{r'}(t+1) + w_i > \mathrm{Tr}$.
- increases by w_i for every task i which was below the threshold $(h_r(t) + w_i \leq \mathrm{Tr})$ and then migrates to a resource r' such that $h_{r'}(t+1) + w_i > \mathrm{Tr}$.

We begin with the following Observation.

Observation 2.15. *Let t be an arbitrary time step and $\phi_r(t) > 0$. Then the number of tasks required to leave r such that $x_r(t+1) < Tr$ is at least*

$$\phi'_r = \left\lceil \frac{\phi_r}{w_{max}} \right\rceil.$$

At this stage we only focus on the potential change during a fixed time-step. We initially calculate the potential change for the case where $w_{min} = w_{max}$ and then generalise it to non-identical weighted tasks.

If $w_{min} = w_{max} = 1$, then

$$\mathbb{E}\left[\Delta\phi_r | i \text{ balls leave}\right] \geq \begin{cases} \frac{\varepsilon}{1+\varepsilon} \cdot i & \text{if } i \leq \phi'_r \text{ balls leave} \\ \phi'_r - i & \text{if } i > \phi'_r \text{ balls leave} \end{cases} \tag{2.10}$$

The number of tasks leaving in the first case is less than ϕ'_r which means that they can only reduce the potential if they land on resources with enough empty capacity. According to Lemma 2.5, at any point in time there exists a fraction of $\varepsilon/(1+\varepsilon)$ resources that are able to store an additional task of any weight $\leq w_{max}$ without violating the threshold. Therefore the potential change in the first case is $\varepsilon/(1+\varepsilon) \cdot i$. In the second case we consider the worst case scenario where on one hand ϕ'_r above threshold tasks leave r and move to another over-loaded resource and on the other hand an additional $(i - \phi'_r)$ tasks located below the threshold leave resource r and end up in an overloaded resource.

In the weighted context, like the uniform case, the potential decreases if $i \leq \phi'_r$ tasks leave and move to resources with enough empty capacity. One can assume that the expected size of a leaving task is x_r/b_r because all tasks can leave with the same probability. In case of $i > \phi'_r$ tasks leaving resource r, similar to the uniform case, we disregard any above threshold tasks in r that move to another resource and become below threshold, hence decreasing the potential.

Therefore, we can bound the potential change as follows.

$$\mathbb{E}\left[\Delta\phi_r | i \text{ balls leave}\right] \geq \begin{cases} \frac{\varepsilon}{1+\varepsilon} \cdot i \cdot \frac{x_r}{b_r} & \text{if } i \leq \phi'_r \text{ balls leave} \\ (\phi'_r - i)\frac{x_r}{b_r} & \text{if } i > \phi'_r \text{ balls leave} \end{cases}$$

According to the definition of ϕ'_r, in the first case, the potential decreases in expectation by $\frac{x_r}{b_r}$ for each leaving task, although the height of this leaving task is below the threshold.

In Lemma 2.16 we find a lower bound for the total expected potential drop during a single time-step.

Lemma 2.16. *Let $\alpha = \frac{\varepsilon}{120(1+\varepsilon)}$ in Algorithm 2.4.1. Assume $Tr = (1+\varepsilon) \cdot W/n + w_{max}$ and $\Phi(X(t)) > 0$. We have*

$$\mathbb{E}\left[\Delta\Phi(t+1) \mid X(t) = x(t)\right] \geq \frac{1}{2} \cdot \frac{\varepsilon}{1+\varepsilon} \cdot \Phi(X(t)).$$

Proof. We consider two different cases, the potential drop and the potential increase. Define $p_r(i)$ as the probability that *exactly* i many tasks leave resource r. As mentioned

before, all the tasks in an over-loaded resource have the same probability to leave. Define the random variable $Y_r(i)$ as follows

$$Y_r(i) = \begin{cases} 1 & \text{if task } i \text{ on resource } r \text{ leaves} \\ 0 & \text{otherwise.} \end{cases}$$

Now we have

$$\begin{aligned}
\mathbb{E}\left[\Delta\phi_r(t+1) \mid X(t) = x(t)\right] &= \sum_{i=0}^{b_r} \mathbb{E}\left[\Delta\phi_r | i \text{ balls leave}\right] \cdot p_r(i) \\
&\geq \sum_{i=1}^{\phi'_r} \frac{\varepsilon}{1+\varepsilon} \cdot i \cdot \frac{x_r}{b_r} \cdot p_r(i) - \sum_{i=\phi'_r+1}^{b_r} i \cdot \frac{x_r}{b_r} \cdot p_r(i) \\
&\geq \frac{x_r}{b_r} \sum_{i=1}^{b_r} \frac{\varepsilon}{1+\varepsilon} \cdot i \cdot p_r(i) - 2\frac{x_r}{b_r} \sum_{i=\phi'_r+1}^{b_r} i \cdot p_r(i) \\
&= \frac{x_r}{b_r} \frac{\varepsilon}{1+\varepsilon} \cdot \mathbb{E}\left[\sum_{i=1}^{b_r} Y_r(i)\right] - 2 \cdot \frac{x_r}{b_r} \sum_{i=\phi'_r+1}^{b_r} i \cdot p_r(i),
\end{aligned}$$

the last equality is valid because all the tasks in an over-loaded resource have the same probability to leave. Since

$$\mathbb{E}\left[Y_r(i)\right] = \alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil \cdot \frac{1}{b_r},$$

then $\mathbb{E}\left[\sum_{i=1}^{b_r} Y_r(i)\right] = \alpha \cdot \left\lceil \frac{\phi_r}{w_{max}} \right\rceil$. In the next step we calculate an upper bound for the potential increase, *i.e.* $\sum_{i=\phi'_r+1}^{b_r} i \cdot p_r(i)$.

In this regard the lesser the value of $\phi'_r \geq 1$ the greater the value of $\sum_{i=\phi'_r+1}^{b_r} i \cdot p_r(i)$. It follows

$$\begin{aligned}
\sum_{i=\phi'_r+1}^{b_r} i \cdot p_r(i) &= \sum_{i=\phi'_r+1}^{b_r} i\binom{b_r}{i}\left(\alpha\left\lceil\frac{\phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i\cdot\left(1-\alpha\left\lceil\frac{\phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^{b_r-i}\\
&\leq \sum_{i=\phi'_r+1}^{b_r} i\binom{b_r}{i}\left(\alpha\left\lceil\frac{\phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i\\
&\leq \sum_{i=\phi'_r+1}^{b_r} i\left(\alpha\frac{e\cdot b_r}{i}\cdot\left\lceil\frac{\phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i\\
&\leq \sum_{i=\phi'_r+1}^{b_r} i\left(\alpha\frac{e\cdot b_r}{\left\lceil\frac{\phi_r}{w_{max}}\right\rceil}\left\lceil\frac{\phi_r}{w_{max}}\right\rceil\cdot\frac{1}{b_r}\right)^i\\
&\leq \sum_{i=2}^{\infty} i(e\alpha)^i \leq 30\alpha^2,
\end{aligned}$$

the second inequality is valid because

$$\binom{b_r}{i} \leq \frac{{b_r}^k}{i!} \leq \left(\frac{b_r\cdot e}{i}\right)^i.$$

Let $\alpha = \frac{\varepsilon}{120(1+\varepsilon)}$ then

$$\begin{aligned}
\mathbb{E}\left[\Delta\phi_r(t+1) \mid X(t)=x(t)\right] &\geq \frac{\varepsilon}{(1+\varepsilon)}\left\lceil\frac{\phi_r}{w_{max}}\right\rceil\cdot\alpha\cdot\frac{x_r}{b_r} - 60\alpha^2\cdot\frac{x_r}{b_r}\\
&\geq \alpha\cdot\frac{\varepsilon}{2(1+\varepsilon)}\cdot\frac{x_r}{b_r}\cdot\frac{\phi_r}{w_{max}}\\
&\geq \alpha\cdot\frac{\varepsilon}{2(1+\varepsilon)}\cdot\frac{w_{min}}{w_{max}}\cdot\phi_r.
\end{aligned}$$

Summing over all resources $r \in [n]$ with $\phi_r \geq 1$ we have

$$\mathbb{E}\left[\Delta\Phi(t+1) \mid X(t)=x(t)\right] \geq \alpha\cdot\frac{\varepsilon}{2(1+\varepsilon)}\cdot\frac{w_{min}}{w_{max}}\cdot\Phi(X(t)).$$

□

Using Lemma 2.16, the main result of this section follows.

Theorem 2.17. *Assume* $Tr = (1+\varepsilon)\cdot W/n + w_{max}$ *and* $\alpha = \frac{\varepsilon}{120\cdot(1+\varepsilon)}$. *Let* T *be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}[T] = 2\cdot\frac{1+\varepsilon}{\alpha\cdot\varepsilon}\cdot\frac{w_{max}}{w_{min}}\cdot\log m.$$

Proof. The result is following from Lemma 2.16 together with Theorem 2.10 (Drift Theorem). □

2.4.3 Tight Threshold

In this section, given Tr $= W/n + w_{max}$ (tight thresholds), we investigate the user-controlled protocols on a complete graph. Theorem 2.18 is the main result of this section.

Theorem 2.18. *Assume* $Tr = W/n + w_{max}$ *and* $\alpha \leq \frac{1}{120n}$. *Let* T *be the time it takes until all tasks are allocated. Then*

$$\mathbb{E}[T] = \frac{2\cdot n}{\alpha}\cdot\frac{w_{max}}{w_{min}}\cdot\log m.$$

Proof. We can trivially argue that at any time-step (before or after task migration) there exists at least one resource that can accept an additional task of any weight ($\leq w_{max}$). As a consequence, the result follows by replacing $\varepsilon/(1+\varepsilon)$ with $1/n$ and setting $\alpha \ll \frac{1}{n}$ in the proof of Theorem 2.17. □

2.5 Conclusion

In this chapter we investigated threshold-based load balancing protocols. We introduced a generalisation of the protocols presented in [2, 3] to weighted tasks. As for resource-controlled protocols, in the above average threshold case our bound is independent of the weights of the tasks and it improves the expected balancing time $O(\mathcal{H}(G)\cdot\log n + \tau(G)\cdot\log m)$ for uniform tasks in [2]. In the tight threshold case, we show that the expected balancing time is $O(\mathcal{H}(G)\cdot\log m)$. This bound is tight and independent of weight and it matches the bound in [2] for uniform tasks. As for user-controlled protocols, we have only considered complete graphs. For the above average threshold case, we show a bound of $O(w_{max}/w_{min}\cdot\log m)$ on the expected balancing time. In tight threshold case, in we show a bound of $O(w_{max}/w_{min}\cdot\frac{\log m}{n^2})$. Our bounds match the bounds in [3] for unit-weight tasks. But for weighted tasks, our bounds include an additional factor of w_{max}/w_{min}.

Chapter 3

Evolutionary Dynamics of the Pull-Moran Process

3.1 Introduction

Evolution in biology is based on the idea that an organism's genes largely determine its observable characteristics and its fitness in a given environment. Organisms that are more fit will tend to produce more offspring and consequently the frequency of these genes within the population will increase [42]. This is the basis of Darwin's theory of *natural selection* - beneficial (or fitter) genes tend to survive over time as they present a higher rate of reproduction. Evolution is the outcome of such mechanisms. *Evolutionary dynamics* is the study of mathematical principles in the evolution of populations. For instance, in evolutionary game theory the competition between individuals can be modelled as strategies in game theory [43–46]. An organism's genetically-determined characteristics and behaviours are like its strategy in a game, its fitness is like its pay-off, and this pay-off depends on the strategies (characteristics) of the organisms with which it interacts [42]. Furthermore, evolutionary dynamics are frequently used to study the evolution of *homogeneous* populations. In this sense a homogeneous population is one in which each member can interact with any other member of the population. Here, observing the interaction between different species and their *birth-death* rate within a population is the matter of interest. Moreover, some game theory ideas like reaching (birth-death) equilibrium or dynamics such as domination (or extinction) of certain species, called the *mutants*, in a given population have been studied. Mutants acquire different fitness rates to the other individuals in the population. In practice, mutants can resemble influential people in a society (or a social network) or they can be seen as a computer in a network that is infected by a virus. Population and evolutionary dynamics

of such systems have been widely studied [47–53], where it is normally assumed that the the population is homogeneous *i.e.* each individual is able interact with all others within the population. In such dynamics, the Moran process is one of the most studied models. In the Moran process [17], given a homogeneous population with a single mutant individual, we would like to know the probability of the mutant descendants taking over the whole population. This probability is called the *fixation probability*. In a Moran process the fixation probability is proportional to the fitness of the mutant. Generally speaking, human societies or social networks are never homogeneous, as certain individuals in central positions may be more influential than others [54]. Lieberman, Hauert and Nowak [18] introduced a new process applied to structured populations or networks. This is the generalisation of the Moran process with homogeneous populations to structured populations where individuals are arranged on a (connected) graph. In this setting the structure of the graph could also play an important role in the fixation process. In some cases fixation can be independent of the fitness due to the structure of a graph, and the fixation state may also be amplified in particular graphs. In this chapter our main goal is to observe the behaviour of the Pull-Moran process (also known as the death-birth update) and show that it is very different from that of the structured Moran process. In the following, after an introduction to the Moran process, we review the related studies that have been conducted in Sections 3.2 and 3.2.2. In Section 3.3 we review the concept of the Pull-Moran process together with its application on star graphs. In the remainder of the chapter we will analyse the Pull-Moran process on different graph classes and prove our main results.

3.1.1 Moran Process

The *Moran process* (also known as the *birth-death* update), first introduced by P. Moran in [17], is an iterative random process applied on a homogeneous population of size N, where all the individuals can interact with each other. In this chapter we use the definition of the Moran process as presented in [18]. Initially the process starts with all identical individuals with fitness 1 except one mutant chosen *u.a.r* from the population that has fitness r. In every iteration an individual is chosen, with a probability proportional to its fitness, for reproduction. Thereafter, the offspring of the chosen individual replaces another individual in the population chosen *u.a.r.* If that individual was a non-mutant, we say it has been *infected* (or become mutated).

Let X_n be the number of infected individuals (mutants) at the step n. According to this definition, $S = \{0, 1, \ldots, N\}$ is the set of all possible states and $i \in S$ is the number of infected vertices. Particularly, $i = 0$ and $i = N$ are called *extinction* and *fixation* states

respectively. For each $i, j \in S$ the probability of moving to state j form i is

$$P_{i,j} = Pr[X_{n+1} = j | X_0 = i_0, \cdots, X_{n-1} = i_{n-1}, X_n = i] = Pr[X_{n+1} = j | X_n = i]$$

and therefore the Moran process forms a Markov chain. Note that, according to the definition of the Moran process, given a state i the next state can only be $i+1$ or $i-1$.

According to [18], the fixation probability of the Moran process, *i.e.* the probability of the whole population becoming mutated, is

$$\rho = \frac{1 - 1/r}{1 - 1/r^N} \tag{3.1}$$

Here, fitter mutants ($r > 1$) have a certain chance to take over the whole population whilst unfit mutants ($r < 1$) are likely to become extinct. However, in both of these cases these outcomes are not guaranteed.

3.2 Related Studies

Similar dynamic processes to the one described above have been investigated before. Models that share the same terms such as infection and individual interactions include the voter model and the rumour spreading models like PUSH and PULL [55–57]. However, these approaches do not follow evolutionary rules, for example fitness, which are considered in the Moran process. The Moran process terminates when it reaches either *fixation*, where all the population become mutants, or *extinction*, where all mutants become extinct. We define $f_r(G)$ is the fixation probability of an undirected connected graph G, with a randomly placed mutant of fitness r. Generally speaking, the fixation probability can be modeled and computed by standard Markov chain techniques. The authors in [18] introduced a Moran process applied on a structured population. They assume that each individual occupies a vertex in a graph so that the interactions between individuals are reduced down to the edges of a connected graph. More formally, individuals are labelled $i = 1, 2, \cdots, N$. Like the Moran process, in every iteration an individual i is chosen randomly based on its fitness. However, it can only choose an individual j *u.a.r* to replace with its offspring if (i, j) is an edge in the population graph. In other words, the offspring of an individual can only take the place of the neighbours of that individual. Note that if the underlying graph is a complete graph then the structured process will become the original Moran process on a homogeneous population with no spatial structure. The goal in the structured process is also to calculate the fixation probability starting with a randomly placed mutant. Furthermore, classifying the types

of graphs with the same fixation probability, or finding general lower (or upper) bounds for arbitrary graphs have been also a matter of discussion. Particularly in [18, 54], the authors initiate the discussion by introducing the *Isothermal Theorem*. This theorem indicates that for all regular graphs (*i.e.* graphs in which all edges have the same degree), the fixation probability is equal to $\rho = \frac{1-1/r}{1-1/r^N}$. This is exactly the same as the fixation probability of the standard Moran process ,*i.e.* with no structure applied. They also consider each vertex to have a temperature based on how often it is replaced by the offspring of other individuals. A vertex is labelled *hot* if it is replaced often and *cold* if it is replaced rarely. The authors in [54] use ρ as a measure to classify arbitrary graphs. They divided graphs into two categories. Suppressor graphs have a fixation probability less than ρ, and amplifier graphs have a fixation probability greater than ρ. Star graphs are proved to be suppressors in [18]. In [58], Broom and Rychtář calculate the fixation probability by solving a linear system of equations or by using numerical methods. In particular they compute the fixation probability for star and path graphs and make a numerical comparison between path and cycle graphs. However, in order to calculate the fixation probability for a graph of size n using exact methods, it is necessary to solve a linear system of 2^n equations. For large n this is not computationally reasonable and therefore this approach is only applicable to a limited number of symmetrical graphs, where the symmetry reduces the number of equations. As for directed graphs, in [18], Lieberman et al. also investigate the fixation probabilities of some layered directed graphs, such as superstar, funnel and metafunnel, where their extreme behaviours would help in finding the fixation probability.

Some other studies have been carried out using approximation methods due to the complexity of the exact method. In [59], the authors use Monte-Carlo techniques to construct fully polynomial randomised approximation schemes for the probability of fixation, where $r \geq 1$, and the probability of extinction, where $r > 0$. Finally they introduce general bounds for the fixation probability of all connected undirected graphs that are $f_r(G) \geq \frac{1}{n}$ and $f_r(G) \leq 1 - \frac{1}{n+r}$. Some other studies focus on the relationship between the structure of a graph and its fixation probability and specifically in finding a method to categorise arbitrary graphs based on their fixation probability. The idea behind this chapter is extracted from [19] where the authors introduce a different terminology for measuring the success of a mutant, *i.e.* fixation, in a structured population. In preference to the random placement of a mutant in the population graph, they count the number of vertices in the graph that can guarantee (*w.h.p.*) the fixation if they are chosen for the initial mutant placement. In this respect, they try to find the graphs that accommodate a relatively large number of strong/weak vertices. We will explain their approach in more detail in Section 3.2.1 since our research is carried out based on their definitions.

3.2.1 Suppressors and Amplifiers of Selection

A new notion of fixation probability is introduced in [19]. Here, the authors consider a mutant placed at a particular vertex v in the graph, *i.e.* the mutant is no longer placed *u.a.r*. Each vertex is then considered to have its own fixation probability based on the overall success of the mutant when starting from that vertex. Then the fixation probability of a graph $G = (V, E)$ is $f_r(G) = \frac{1}{n}\sum_{v\in V} f_r(v)$ where $n = |V|$. Based on their definition a graph G is a $(h(n), g(n))$-*selective amplifier* if there exists at least $h(n)$ vertices with $f_r(v) \geq 1 - \frac{c(r)}{g(n)}$ for an appropriate function $c(r)$ of r. Similarly a graph G is a $(h(n), g(n))$-*selective suppressor* if there exist at least $h(n)$ vertices with $f_r(v) \leq \frac{c(r)}{g(n)}$ for an appropriate function $c(r)$ of r. Based on these new measures of fixation, they find a particular class of graphs, called *urchin* graphs, which exhibit interesting behaviour. An urchin consists of a clique of size n and an independent set of size n where there is a perfect matching between the clique and the independent set. Urchin graphs are discussed further in Section 3.6.1. They show that urchin graphs are $(\Theta(n), n)$-*selective amplifiers*, otherwise known as *strong selective amplifiers*. They also introduce another group of graphs, called $\phi(n)$-urchin, which fit into the suppressors category. In particular:

Theorem 3.1 (Theorem 5. [19]). *For every function $\phi(n)$, where $\phi(n) = \Omega(1)$ and $\phi(n) \leq \sqrt{n}$, then family of $\phi(n)$-urchin graphs is a class of $(\frac{n}{\phi(n)+1}, \frac{n}{\phi(n)})$-*selective suppressor

Furthermore, they improved both the lower bound and upper bound first introduced in [59]. Particularly, they show that there is no graph that belongs to the class of $(\Theta(n), \frac{c(r)}{n^{3/4+\varepsilon}})$-*selective amplifiers*. In other words, there is no graph G with $f_r(G) < 1 - \frac{c(r)}{n^{3/4+\varepsilon}}$. For the lower bound, they introduce the *Thermal Theorem* in which they show that for any vertex v of an undirected graph G and $r > 1$, the fixation probability $f_r(v)$ has a lower bound of $\frac{r-1}{r+\deg v/\deg_{min}}$ where $\deg v$ is the degree of vertex v and $\deg_{min}$ is the minimum degree in G. This new lower bound is better than the bound presented in the *isothermal theorem* in [18] specifically for large n.

3.2.2 The Pull-Moran (Death-Birth) Protocol

Our focus in this chapter is mainly on a type of evolutionary dynamics called the *death-birth* update, which we term the Pull-Moran protocol. Here, at each time step, a random individual is selected to die. Thereafter, the vacant place is filled by one of its neighbours which is chosen with a probability proportional to the fitness. In the rumour spreading setting the Moran process, *i.e.* replacing a neighbour with the offspring of the

chosen individual, resembles the PUSH strategy whilst the death-birth protocol resembles the PULL strategy. This is the idea behind naming the latter as the PULL-Moran protocol. In [60], the authors show that the star graph, which is a selective amplifier under the Moran process, becomes a selective suppressor under the death-birth protocol. Ohtsuki and Nowak in [61] study evolutionary game dynamics in structured populations. Here, the fitness of an individual is locally determined from interactions with its neighbours in the graph. Each player is a vertex of the graph where birth-death, death-birth and imitation are the update rules. They also introduce a system of ordinary differential equations that show how the expected frequency of each strategy in a game on a graph changes over time. Kaveh, Komarova and Kohandel [62] show how the isothermal theorem can be formulated for death-birth processes. Finally, Hindersin and Traulsen [63] show that almost any undirected random graph is a selective amplifier for the birth-death process, and the same class of graphs are selective suppressors for the death-birth process.

3.2.3 Our Contribution

Instead of analysing the influence of individuals with different fitnesses, we focus on the likelihood that an individual is influenced (or killed) by another in the network. We call this the PULL-Moran process. If one of the important motivations behind the structured Moran process is to analyse the behaviours in social networks, where some individuals are more influential than others, then studying how different individuals are influenced by their circle of friends is also of great importance. In the PULL-Moran process global information, such as the total number of mutants in the system, is not required. Furthermore, motivated by the study conducted in [19], we analyse the behaviour of the PULL-Moran process subject to different classes of graph including regular graphs and selective suppressors. Our analysis is divided into three different sections; the non-existence of the graphs that have a very large total fixation probability, the relativity of the fixation probabilities of a vertex with respect to the standard Moran and the PULL-Moran processes, and finally the relativity of the total fixation probabilities of a regular graph with respect to the standard Moran and the PULL-Moran processes. In Section 3.5 we first show that a class of strong universal amplifiers does not exist. Specifically, in Theorem 3.7 we show that there is no graph G such that $f_r(G) \geq 1 - \frac{c(r)}{n}$. In Section 3.6.1 we will show that urchin graphs are strong selective suppressors. We then, in Theorem 3.9, show that for an arbitrary node in an arbitrary graph G, the upper bound on the fixation probability in the PULL-Moran is related to the lower bound on the fixation probability in the standard Moran. Particularly, in Corollary 3.10 we show that any class of selective amplifiers in the (standard) Moran process belongs in the

selective suppressor class under the PULL-Moran process. In Observation 3.11, again we show that the upper bound on the fixation probability of any vertex in a regular graph G under the standard Moran approach is related to the lower bound of the same vertex under the PULL-Moran approach. In Theorem 3.12 we improve the results in Observation 3.11 and show that the total fixation probability of a regular graph in the PULL-Moran process is upper bounded by the total fixation probability of the same graph under the standard Moran process. Finally, in Section 3.8, we introduce a class of graphs called punk graphs and show that they are practically selective amplifiers under the PULL-Moran process. To the best of our knowledge punk graphs are the first known selective amplifiers in this framework.

3.3 Model and Definitions

Like the structured Moran process, the PULL-Moran process is also applied on a structured population. Similarly, it is initiated by placing a mutant with fitness r in the population graph and it is terminated by fixation or extinction of the mutant. However, the PULL-Moran's iterative protocol is very different from that of the standard Moran process. Here, at each time step a vertex is chosen *u.a.r* and then it will be replaced by the offspring of one of the neighbouring vertices chosen randomly based on their fitnesses (see Algorithm 3.3.1). Comparing to the standard Moran, the PULL-Moran approach resembles the pull version of the standard Moran. Following the same definitions as in [19], our goal is to find upper and lower bounds on $f_r(G)$ for an arbitrary graph G and also to discover other types of graph that represent known amplifier or suppressor classes. Before we define our model, we need to review some of the key definitions

Algorithm 3.3.1 The PULL-Moran

while not Fixation nor Extinction **do**
  Choose vertex v in G *u.a.r*
  Choose one of the neighbours (of v) randomly based on their fitnesses
  Replace v with the chosen neighbour

introduced in [19] in more detail.

According to the nature of the PULL-Moran process, as for the structured Moran at each time step there are three different possibilities; the possibility that one of the normal vertices becomes *infected* (*i.e.* it turns into a mutant), the possibility that a Mutant becomes disinfected and finally the possibility that the number of mutants remain unchanged. For each graph $G = (V, E)$ let $S_t \subseteq V$ be the set of infected vertices (mutants) at the end of time step t then, based on the above argument, we have

$S_{t+1} := S_t \setminus \{u\}$ or $S_{t+1} := S_t \cup \{v\}$ or $S_{t+1} := S_t$ where $u \in S_t$ and $v \in V \setminus S_t$. Also we define the constant r to be the fitness of the mutant.

Definition 3.2. *The fixation probability of the set $S \subseteq V$ denoted by $f_r(S)$ is the probability of reaching the fixation having set all $v \in S$ as mutant at the beginning.*

Based on the above definition, $f_r(\varnothing) = 0$, $f_r(V) = 1$.

Definition 3.3. *Let $\mathcal{G}$ be a class of undirected graphs. If there exists an $n_0 \in \mathbb{N}$, $r_0 > 1$ and some function $c(r)$, such that for every graph $G \in \mathcal{G}$ with $n \geq n_0$ vertices and for every $r > r_0$*

- *if $f_r(G) \geq 1 - \frac{c(r)}{g(n)}$ then $\mathcal{G}$ is a class of $g(n)$-universal amplifiers*
- *if there exists a subset S of at least $h(n)$ vertices of G, such that $f_r(v) \geq 1 - \frac{c(r)}{g(n)}$ for every vertex $v \in S$, then $\mathcal{G}$ is a class of $(h(n), g(n))$-selective amplifiers.*

Definition 3.4. *Let $\mathcal{G}$ be an infinite class of undirected graphs. If there exists functions $c(r)$ and $n_0(r)$, such that for every $r > 1$ and for every graph $G \in \mathcal{G}$ with $n \geq n_0(r)$ vertices*

- *if $f_r(G) \leq \frac{c(r)}{g(n)}$ then $\mathcal{G}$ is a class of $g(n)$-universal suppressors*
- *if there exists a subset S of at least $h(n)$ vertices of G, such that $f_r(v) \leq \frac{c(r)}{g(n)}$ for every vertex $v \in S$, then $\mathcal{G}$ is a class of $(h(n), g(n))$-selective suppressors.*

Definition 3.5. *If $\mathcal{G}$ is a class of n-universal ($(\Theta(n), n)$)-selective) suppressors then $\mathcal{G}$ is called a class of strong universal (selective) suppressors. Also if $\mathcal{G}$ is a class of n-universal ($(\Theta(n), n)$)-selective) amplifiers then $\mathcal{G}$ is labelled as a class of strong universal (selective) amplifiers.*

As an initial example, we apply the Pull-Moran to more symmetric graphs such as the star graph. Here, finding the exact formulation of the fixation probability is not the goal, instead we are interested in calculating a lower bound on the fixation probability of the infected center of the star using methods that reduce the corresponding stochastic process to a less complex one. Influenced by the amplifying feature of the star, we then introduce the first known class of amplifiers under Pull-Moran process in Section 3.8.

3.4 The Pull-Moran on Star

When calculating the fixation probability of a star, we deal with two different types of vertices; the center of the star and the leaves. Therefore, we need to consider two

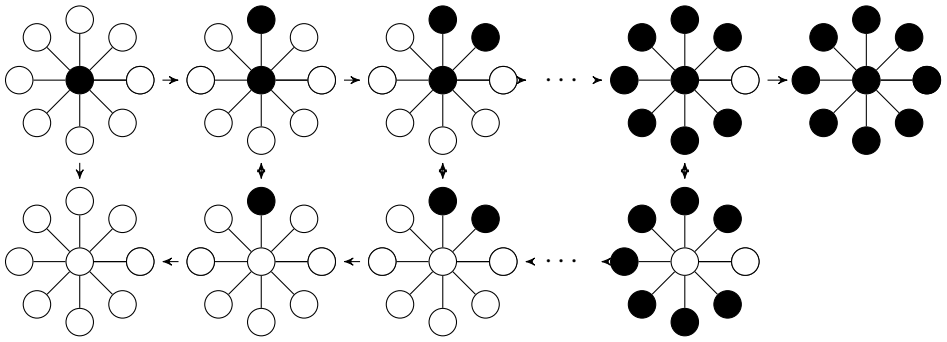

FIGURE 3.1: PULL-Moran chain for star

different (Markov) processes; the process in which the center is infected and the process in which the center is not infected (see Figure 3.1).

Let $Q_{i,\text{on}}$ be the fixation probability given that i infected vertices (mutants) exist on the graph along with an infected center (in total there are i+1 infected vertices) and let $Q_{i,\text{off}}$ be the fixation probability given i infected vertices with a non-infected center (see Figure 3.2). We assume that $P_{\text{on}}^{i\to i+1}$ is the probability of moving from the state with i infected leaves to the state with $i+1$ infected leaves whilst the center is infected throughout the transformation. Similarly let $P_{\text{off}}^{i\to i-1}$ be the probability of moving from a state with i infected leaves and a non-infected center to a state with $i-1$ infected leaves and a non-infected center. Furthermore, we assume $P_{\text{on}\to\text{off}}^{i}$ is the probability of moving from the state with i infected leaves and an infected center to a state with i infected leaves and a non-infected center and finally $P_{\text{off}\to\text{on}}^{i}$ is the probability of moving from a state with i infected leaves and a non-infected center to a state with the same number of infected leaves and an infected center. Therefore, for a star $G(V,E)$ with $|V| = N+1$, $|E| = N$ we have:

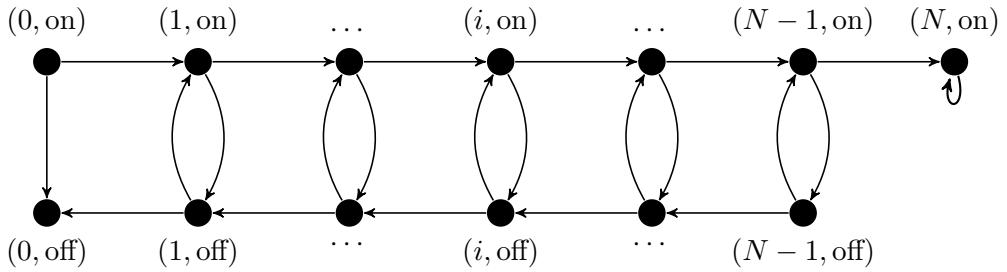

FIGURE 3.2: PULL-Moran transitions in star

$$Q_{i,\text{on}} = P_{\text{on}\to\text{off}}^{i} Q_{i,\text{off}} + P_{\text{on}}^{i\to i+1} Q_{i+1,\text{on}} + (1 - P_{\text{on}\to\text{off}}^{i} - P_{\text{on}}^{i\to i+1}) Q_{i,\text{on}} \tag{3.2}$$

$$Q_{i,\text{off}} = P^{i}_{\text{off}\to\text{on}} Q_{i,\text{on}} + P^{i\to i-1}_{\text{off}} Q_{i-1,\text{off}} + (1 - P^{i}_{\text{off}\to\text{on}} - P^{i\to i-1}_{\text{off}}) Q_{i,\text{off}} \tag{3.3}$$

Where

$$P^{i\to i+1}_{\text{on}} = \frac{N-i}{N+1}$$

$$P^{i}_{\text{on}\to\text{off}} = \frac{1}{N+1} \cdot \frac{N-i}{ri+N-i}$$

$$P^{i}_{\text{off}\to\text{on}} = \frac{1}{N+1} \cdot \frac{ri}{ri+N-i}$$

$$P^{i\to i-1}_{\text{off}} = \frac{i}{N+1}.$$

Solving Eqn. (3.2) and Eqn. (3.3) for $Q_{i,\text{off}}$ and $Q_{i,\text{off}}$ respectively yields

$$Q_{i,\text{on}} = \frac{1}{\alpha_i + 1} Q_{i,\text{off}} + \frac{\alpha_i}{1+\alpha_i} Q_{i+1,\text{on}} \tag{3.4}$$

$$Q_{i,\text{off}} = \frac{\alpha_i}{\alpha_i + r} Q_{i-1,\text{off}} + \frac{r}{\alpha_i + r} Q_{i,\text{on}} \tag{3.5}$$

where $\alpha_i = ri + N - i$.

Calculating the exact fixation probability for a star graph is not as straightforward as we mentioned before. However, we can utilize some methods that will help in finding the upper and lower bounds on the fixation probability of certain vertices. Particularly, in this section, we are interested in finding a lower bound for the fixation probability of a process initiated with an infected center.

In order to lower bound the fixation probability of a star given an infected center, we need to adopt a policy in favour of non-infected nodes. In other words, if we manipulate the process in a way that the non-mutant vertices are more beneficial than the mutants then the new fixation probability of the mutants will be reduced when compared to the original process. Here, we assume that the transition from a state with i infected leaves and a non-infected center (i, off), instead of going to the state with $i-1$ infected leaves and a non-infected center $(i-1, \text{off})$, goes directly to the state $(0, \text{off})$, see the difference between Figure 3.3 and Figure 3.2. More formally, Eqn. (3.5) will change to

$$Q_{i,\text{off}} = 0 + \frac{r}{\alpha_i + r} Q_{i,\text{on}}. \tag{3.6}$$

Therefore from Eqn. (3.4) and Eqn. (3.6) it follows that

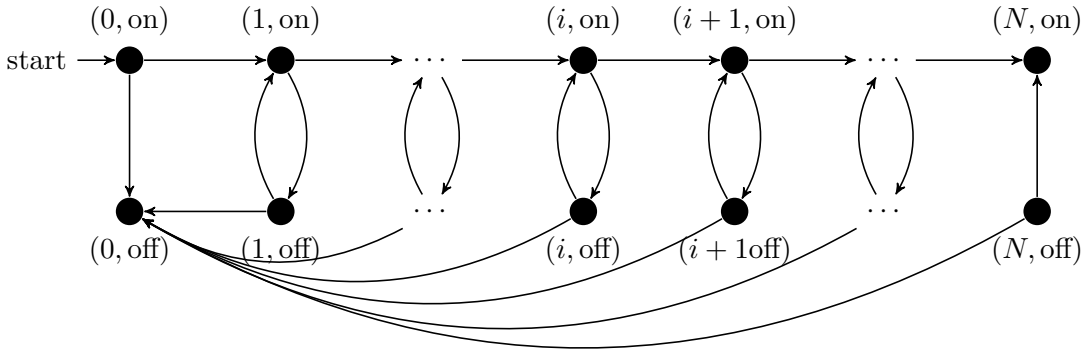

FIGURE 3.3: Manipulated PULL-Moran chain in star

$$Q_{i,\text{on}} = \frac{\alpha_i}{1+\alpha_i} Q_{i+1,\text{on}} + \frac{1}{\alpha_i + 1}\left(\frac{r}{r+\alpha_i}\right) Q_{i,\text{on}} \tag{3.7}$$

solving for $Q_{i,\text{on}}$,

$$Q_{i,\text{on}} = \frac{\alpha_i + r}{\alpha_i + r + 1} Q_{i+1,\text{on}}. \tag{3.8}$$

By induction we have

$$\begin{aligned} Q_{0,\text{on}} &= \left(\frac{\alpha_0 + r}{\alpha_0 + r + 1} \cdot \frac{\alpha_1 + r}{\alpha_1 + r + 1} \cdots \frac{\alpha_{N-1} + r}{\alpha_{N-1} + r + 1}\right) Q_{N,\text{on}} \\ &= \frac{\alpha_0 + r}{\alpha_0 + r + 1} \cdot \frac{\alpha_1 + r}{\alpha_1 + r + 1} \cdots \frac{\alpha_{N-1} + r}{\alpha_{N-1} + r + 1} \\ &= \frac{N + r}{N + r + 1} \cdot \frac{(r-1) + N + r}{(r-1) + N + r + 1} \cdots \frac{(r-1)(N-1) + N + r}{(r-1)(N-1) + N + r + 1} \\ &\geq \left(\frac{N}{N+1}\right)^N = \frac{1}{\left(\frac{N+1}{N}\right)^N} \geq \frac{1}{e} \end{aligned}$$

3.5 Non-existence of Strong Universal Amplifiers

In the following we will present our main results regarding the so-called PULL-Moran process. Our analysis is divided into three different sections, the non-existence of the graphs that have a very large total fixation probability, the relativity of the fixation probabilities of a vertex with respect to the standard Moran and the PULL-Moran processes, and finally the relativity of the total fixation probabilities of a regular graph with respect to the standard Moran and the PULL-Moran processes. According to Definition 3.5, $\mathcal{G}$ is a class of *strong universal amplifiers* if $f_r(G) \geq 1 - \frac{c(r)}{n}$. In [19], the authors show that a class of strong universal amplifiers cannot exist (Theorem 3.6).

Theorem 3.6 (Theorem 1. [19]). *For any function $g(n) = \Omega(n^{\frac{3}{4}+\epsilon})$ for some $\epsilon > 0$, there exists no graph class $\mathcal{G}$ of $g(n)$-universal amplifiers for any $r > r_0 = 1$.*

Following the same approach in [19], we want to find an upper bound for universal amplifiers when $r > 1$. In other words, we want to show that for all classes of universal amplifiers (Definition 3.5), we have $f_r(G) \leq 1 - \frac{1}{n^{1-\delta}}$.

Theorem 3.7. *For any function $g(n) = \Omega(n^{1-\delta})$ for some $\delta > 0$, under the* Pull-Moran *process, there exists no graph within the class of $g(n)$-universal amplifiers given $r > r_0 = 1$.*

Proof. We assume that $\mathcal{G}$ is a class of $g(n)$-universal amplifiers. Therefore for every $G(V, E) \in \mathcal{G}$ with $|V| = n > n_0$ vertices, the fixation probability of G is $f_r(G) \geq 1 - \frac{c(r)}{g(n)} \geq 1 - \frac{1}{n^{1-\delta}}$ for $r > 1$ and a $\delta < 1$. From every $v \in V$ we calculate an upper bound for $f_r(v)$ by assuming that fixation is obtained if the process infected two vertices given v itself is infected. More formally, for every vertex v of graph $G(V, E)$ under the Pull-Moran approach, let S_1 be the state where v is the only infected vertex and $S_2(u)$ be the state where v, u $(u \in N(v))$ are infected. Also let $p_2(u)$ be the probability of transition from state S_1 to $S_2(u)$ and p_1 is the probability of transition from state S_1 to the state $S_0 = \varnothing$, where no vertex is infected. Then based on these definitions

$$f_r(v) = \sum_{u \in N(v)} p_2(u) f_r(S_2(u)) + p_1 f_r(S_0) + \left(1 - p_1 - \sum_{u \in N(v)} p_2(u)\right) f_r(v).$$

Solving for $f_r(v)$

$$\begin{aligned} f_r(v) &= \frac{\sum_{u \in N(v)} p_2(u) f_r(S_2(u)) + p_1 f_r(S_0)}{\sum_{u \in N(v)} p_2(u) + p_1} \\ &= \frac{\sum_{u \in N(v)} p_2(u) f_r(S_2(u))}{\sum_{u \in N(v)} p_2(u) + p_1} \end{aligned} \tag{3.9}$$

as $f_r(S_0) = 0$.

Since

$$\sum_{u \in N(v)} p_2(u) = \sum_{u \in N(v)} \left(\frac{1}{n} \cdot \frac{r}{r + \deg u - 1}\right)$$

and

$$p_1 = \frac{1}{n} \cdot 1,$$

then we have

$$f_r(v) = \frac{\sum_{u\in N(v)} p_2(u) f_r(S_2(u))}{\sum_{u\in N(v)} p_2(u) + p_1} \leq \frac{\sum_{u\in N(v)} p_2(u)\cdot 1}{\sum_{u\in N(v)} p_2(u) + p_1} \tag{3.10}$$

Therefore

$$f_r(v) \leq \frac{\frac{r}{n}\sum_{u\in N(v)} \frac{1}{r-1+deg(u)}}{\frac{r}{n}\sum_{u\in N(v)} \frac{1}{r-1+deg(u)} + \frac{1}{n}} = 1 - \frac{1}{r\sum_{u\in N(v)} \frac{1}{r-1+deg(u)} + 1} \tag{3.11}$$

We define $R_v = \sum_{u\in N(v)} \frac{1}{r-1+deg(u)}$. It follows

$$f_r(G) = \frac{1}{n}\sum_{v\in V} f_r(v) \leq \frac{1}{n}\left(n - \sum_{v\in V} \frac{1}{r\cdot R_v + 1}\right). \tag{3.12}$$

Following the definition

$$\frac{1}{n}\left(n - \sum_{v\in V} \frac{1}{r\cdot R_v + 1}\right) \geq 1 - \frac{1}{n^{1-\delta}}. \tag{3.13}$$

Hence

$$\sum_{v\in V} \frac{1}{r\cdot R_v + 1} \leq n^{\delta} \tag{3.14}$$

To reach the contradiction, we need to show that Eqn. (3.14) is not valid.

If $Q_v = \sum_{u\in N(v)} \frac{1}{deg(u)}$ then

$$\sum_{v\in V} R_v \leq \sum_{v\in V} Q_v = n. \tag{3.15}$$

Now, considering Eqn. (3.14) and Eqn. (3.15), We will prove that $\sum_{v\in V} \frac{1}{r\cdot R_v+1}$ is minimum if $R_v = R_w$ for every $v, w \in V$. We reach the proof by contradiction. Assume there exist $v_1, v_2 \in V$ with $R_{v_1} \neq R_{v_2}$ such that

$$\begin{aligned}&\sum_{v\in V\setminus\{v_1,v_2\}}\frac{1}{r\cdot R_v+1}+\frac{1}{r\cdot R_{v_1}+1}+\frac{1}{r\cdot R_{v_2}+1}\\&\leq\sum_{v\in V\setminus\{v_1,v_2\}}\frac{1}{r\cdot R_v+1}+\frac{2}{r\cdot(R_{v_1}+R_{v_2})/2+1}.\end{aligned}\tag{3.16}$$

Hence

$$\frac{1}{r\cdot R_{v_1}+1}+\frac{1}{r\cdot R_{v_2}+1}\leq\frac{2}{r\cdot(R_{v_1}+R_{v_2})/2+1}.$$

Therefore $r^2(R_{v_1}-R_{v_2})^2\leq 0$ that is a contradiction with $R_{v_1}\neq R_{v_2}$.

Since $\sum_{v\in V}R_v\leq\sum_{v\in V}Q_v=n$ then $R_v\leq 1$ for every $v\in V$ and therefore

$$\sum_{v\in V}\frac{1}{r\cdot R_v+1}\geq\frac{n}{r+1}$$

Hence, from Eqn. (3.14), it follows that

$$\frac{n}{r+1}\leq n^{\delta}$$

which contradicts our assumption given a sufficiently small δ.

□

3.6 The Bridge Thermal Theorem and Strong Suppressors

Considering the Pull-Moran as the running process, we initially reintroduce urchin graphs this time as a class of strong selective suppressors. Urchin graphs were primarily introduced in [19] as a class of strong amplifier in the context of structured Moran processes. Thereafter, motivated by the contrasting behaviour of the aforementioned processes on the urchin graph, we build our main claim of this section in the *Bridge Thermal Theorem*.

3.6.1 A Class of Strong Selective Suppressor

According to the Definition 3.4 in [19], $\mathcal{G}$ is a class of $(h(n),g(n))$-selective suppressors if there exists a subset S of at least $h(n)$ vertices of $G\in\mathcal{G}$, such that $f_r(v)\leq\frac{c(r)}{g(n)}$ for every vertex $v\in S$. The authors in [19] also introduced a class of graphs called *urchin*, which consist of a clique of size n and an independent set of size n where there is a perfect

matching between the clique and independent set, see Figure 3.4. Applying the standard structured Moran they show that urchin graphs are $(\Theta(n), n)$-*selective amplifiers*, also known as *strong selective amplifiers*. Applying the PULL-Moran process on the same category of graphs, we will show that urchin graphs are *strong suppressors* in our case.

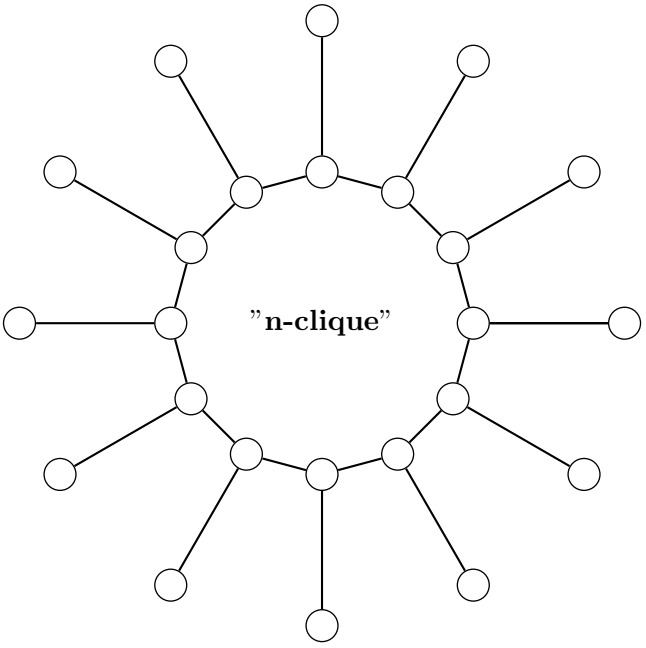

FIGURE 3.4: The urchin graph with $2n$ vertices

For ease of reference, a vertex in the clique section of the urchin is called a *clique vertex* and a vertex located in the independent set is called a *tail vertex*. In the following we show that $f_r(v) \leq \frac{c(r)}{n}$ for a tail vertex v, which would prove that urchin graphs are strong suppressors in this case.

Observation 3.8. *Urchin graphs are strong selective suppressors.*

Proof. We will try to upper bound the fixation probability of the urchin given an infected tail. Assume v_t is a tail vertex of an urchin graph G of size $2n$ (Figure 3.4). Let $f_r(v_t)$ be the fixation probability starting from the infected vertex v_t. Also let S_0 be the state where all the vertices are uninfected (extinction). Define S_1 as the state where v_t and its corresponding (clique vertex) neighbour are infected. If p_1 is the probability of transition from the state where v_t is infected to the state S_1 and p_2 is the probability of a transition from the state with infected v_t to the state S_0, then

$$f_r(v_t) = p_1 f_r(S_1) + p_2 f_r(S_0) + (1 - p_1 - p_2) f_r(v_t).$$

Solving for $f_r(v)$ we obtain

$$f_r(v_t) = \frac{p_1 f_r(S_1) + p_2 f_r(S_0)}{p_1 + p_2} \tag{3.17}$$

where $p_1 = \frac{1}{2n} \cdot \frac{r}{n-1+r}$ and $p_2 = \frac{1}{2n} \cdot 1$.

From Eqn. (3.17) and the fact that $f_r(S_0) = 0$ it follows

$$\begin{aligned} f_r(v_t) &= \frac{p_1}{p_1 + p_2} \cdot f_r(S_1) \\ &= \frac{\frac{1}{2n} \cdot \frac{r}{n-1+r}}{\frac{1}{2n} \cdot \frac{r}{n-1+r} + \frac{1}{2n}} \cdot f_r(S_1) \\ &= \frac{r}{2r + n - 1} \cdot f_r(S_1) \\ &\leq \frac{r}{n-1} \cdot 1. \end{aligned} \tag{3.18}$$

This completes the proof of the observation. □

3.6.2 The Bridge Thermal Theorem

As was mentioned in the previous section, in an urchin graph a tail vertex has a high fixation probability under the standard Moran regulations whilst the same vertex can play the role of a strong suppressor in the Pull-Moran framework. This is the motivation behind the following theorem. In this section we introduce the *The Bridge Thermal Theorem* where we show that, in all vertices, the behaviour of the Pull-Moran depends on the behaviour of the standard Moran. Specifically, we observe that a vertex v can act very differently under the two known protocols *i.e.* the Moran and the Pull-Moran protocols. Inspired by the Thermal theorem [19], which shows a general upper bound on the fixation probability, and the fact that we are bridging the Moran and Pull-Moran processes, we call this *The Bridge Thermal Theorem*. Let $f_r(v)$ ($f'_r(v)$) be the fixation probability of vertex v when applying the standard Moran (Pull-Moran) process. We want to show that the upper bound of $f'_r(v)$ is related to the lower bound of $f_r(v)$.

Theorem 3.9 (Bridge Thermal Theorem)**.** *Let $f_r(v)$ and $f'_r(v)$ be the fixation probabilities of vertex $v \in V$ by applying the standard Moran and the Pull-Moran on a graph $G(V, E)$ respectively. For every vertex v and $r > 1$ we have $f'_r(v) \leq r^2 \cdot q$ if $f_r(v) \geq 1 - q$ where $q \in [0, 1]$.*

Proof. We prove this theorem by contradiction. Assume

$$f'_r(v) > r^2 \cdot q. \tag{3.19}$$

Let S_1 be the state that v is the only infected vertex and $S_2(u)$ be the state that v, u, where $u \in N(v)$, are infected, subject to the standard Moran process. Also let $p_2(u)$ be the probability of transition from state S_1 to $S_2(u)$ and p_1 is the probability of transition from state S_1 to the state $S_0 = \varnothing$, where no vertex is infected. Then based on the definition

$$f_r(v) = \sum_{u \in N(v)} p_2(u) f_r(S_2(u)) + p_1 f_r(S_0) + \left(1 - p_1 - \sum_{u \in N(v)} p_2(u)\right) f_r(v).$$

Solving for $f_r(v)$

$$\begin{aligned} f_r(v) &= \frac{\sum_{u \in N(v)} p_2(u) f_r(S_2(u)) + p_1 f_r(S_0)}{\sum_{u \in N(v)} p_2(u) + p_1} \\ &= \frac{\sum_{u \in N(v)} p_2(u) f_r(S_2(u))}{\sum_{u \in N(v)} p_2(u) + p_1} \end{aligned} \tag{3.20}$$

as $f_r(S_0) = 0$.

Since

$$\sum_{u \in N(v)} p_2(u) = \sum_{u \in N(v)} \left(\frac{r}{r+n-1} \cdot \frac{1}{\deg v}\right) = \frac{r}{r+n-1}$$

and

$$p_1 = \sum_{u \in N(v)} \frac{1}{r+n-1} \cdot \frac{1}{\deg u},$$

then from Eqn. (3.20) we have

$$\begin{aligned} f_r(v) &= \frac{\sum_{u \in N(v)} p_2(u) f_r(S_2(u))}{\sum_{u \in N(v)} p_2(u) + p_1} \\ &\leq \frac{\sum_{u \in N(v)} p_2(u) \cdot 1}{\sum_{u \in N(v)} p_2(u) + p_1} \\ &= \frac{\frac{r}{r+n-1}}{\frac{r}{r+n-1} + \frac{1}{r+n-1} \sum_{u \in N(v)} \frac{1}{\deg u}} \\ &= \frac{r}{r + \sum_{u \in N(v)} \frac{1}{\deg u}}. \end{aligned} \tag{3.21}$$

Similarly, for every vertex v of graph $G(V, E)$ under the PULL-Moran approach, let S_1' be the state where v is the only infected vertex and $S_2'(u)$ be the state where v, u ($u \in N(v)$) are infected. Also let $p_2'(u)$ be the probability of transition from state S_1' to $S_2'(u)$ and p_1' is the probability of transition from state S_1' to the state $S_0' = \varnothing$, where no vertex is infected. Then based on these definitions

$$f'_r(v) = \sum_{u\in N(v)} p'_2(u) f'_r(S'_2(u)) + p'_1 f'_r(S'_0) + \left(1 - p'_1 - \sum_{u\in N(v)} p'_2(u)\right) f'_r(v).$$

Solving for $f'_r(v)$

$$\begin{aligned} f'_r(v) &= \frac{\sum_{u\in N(v)} p'_2(u) f'_r(S'_2(u)) + p'_1 f'_r(S'_0)}{\sum_{u\in N(v)} p'_2(u) + p'_1} \\ &= \frac{\sum_{u\in N(v)} p'_2(u) f'_r(S_2(u))}{\sum_{u\in N(v)} p'_2(u) + p'_1} \end{aligned} \tag{3.22}$$

as $f_r(S'_0) = 0$.

Since

$$\sum_{u\in N(v)} p'_2(u) = \sum_{u\in N(v)} \left(\frac{1}{n} \cdot \frac{r}{r + \deg u - 1}\right)$$

and

$$p'_1 = \frac{1}{n} \cdot 1,$$

then from Eqn. (3.22) we have

$$\begin{aligned} f'_r(v) &= \frac{\sum_{u\in N(v)} p'_2(u) f'_r(S'_2(u))}{\sum_{u\in N(v)} p'_2(u) + p'_1} \\ &\le \frac{\sum_{u\in N(v)} p'_2(u) \cdot 1}{\sum_{u\in N(v)} p'_2(u) + p'_1} \\ &= \frac{\sum_{u\in N(v)} \frac{r}{r+\deg u-1}}{1 + \sum_{u\in N(v)} \frac{r}{r+\deg u-1}}. \end{aligned} \tag{3.23}$$

According to the assumption and From Eqn. (3.21)

$$1 - q \le f_r(v) \le \frac{r}{r + \sum_{u\in N(v)} \frac{1}{\deg u}}. \tag{3.24}$$

Therefore

$$q \ge 1 - \frac{r}{r + \sum_{u\in N(v)} \frac{1}{\deg u}} = \frac{\sum_{u\in N(v)} \frac{1}{\deg u}}{r + \sum_{u\in N(v)} \frac{1}{\deg u}} \tag{3.25}$$

From Eqn. (3.19), Eqn. (3.23) and Eqn. (3.25) follows

$$\frac{1}{r^2} \cdot \frac{\sum_{u\in N(v)} \frac{r}{r+\deg u-1}}{1 + \sum_{u\in N(v)} \frac{r}{r+\deg u-1}} \ge \frac{1}{r^2} \cdot f'_r(v) > q \ge \frac{\sum_{u\in N(v)} \frac{1}{\deg u}}{r + \sum_{u\in N(v)} \frac{1}{\deg u}}. \tag{3.26}$$

Hence

$$\begin{aligned} &\frac{1}{r^2}\cdot\frac{\sum_{u\in N(v)}\frac{r}{r+\deg u-1}}{1+\sum_{u\in N(v)}\frac{r}{r+\deg u-1}} > \frac{\sum_{u\in N(v)}\frac{1}{\deg u}}{r+\sum_{u\in N(v)}\frac{1}{\deg u}} \\ \Rightarrow &r^2\cdot\left(1+\frac{1}{\sum_{u\in N(v)}\frac{r}{r+\deg u-1}}\right) < 1+\frac{r}{\sum_{u\in N(v)}\frac{1}{\deg u}} \end{aligned} \tag{3.27}$$

since $\sum_{u\in N(v)}\frac{r}{r+\deg u-1} \leq r\cdot\sum_{u\in N(v)}\frac{1}{\deg u}$ we have

$$\begin{aligned} &r^2\cdot\left(1+\frac{1}{r\cdot\sum_{u\in N(v)}\frac{1}{\deg u}}\right) < 1+\frac{r}{\sum_{u\in N(v)}\frac{1}{\deg u}} \\ \Rightarrow &r^2+\frac{r}{\sum_{u\in N(v)}\frac{1}{\deg u}} < 1+\frac{r}{\sum_{u\in N(v)}\frac{1}{\deg u}} \\ \Rightarrow &r^2<1. \end{aligned} \tag{3.28}$$

Which contradicts with our assumption $r > 1$. □

In the next corollary we generalise the results in [63] to arbitrary graphs and show that all strong amplifiers of the structured Moran process are actually playing the role of strong suppressors subject to the Pull-Moran process.

Corollary 3.10. *If $\mathcal{G}$ is a class of $(h(n), g(n))$-selective amplifiers in the standard Moran process then $\mathcal{G}$ is a class of $(h(n), g(n))$-selective suppressors under the* Pull-*Moran process.*

Proof. If $\mathcal{G}$ is a class of $(h(n), g(n))$-selective amplifiers then for every graph $G(V, E) \in \mathcal{G}$ with $n \geq n_0$ vertices and for every $r > r_0$, where $n_0 \in \mathbb{N}$, $r_0 > 1$ and $c(r)$ is a function of r, there exists a subset $S \subseteq V$ of at least $h(n)$ vertices such that $f_r(v) \geq 1 - \frac{c(r)}{g(n)}$ for every vertex $v \in S$. From Theorem 3.9 it follows that $f'_r(v) \leq \frac{r^2\cdot c(r)}{g(n)}$ for every vertex $v \in S$, subject to the Pull-Moran protocol. This completes the proof as $\mathcal{G}$ is a class of $(h(n), g(n))$-selective suppressors base, see Definition 3.3. □

3.7 The Pull-Moran on Regular graphs

As in the previous section, we first analyse the fixation probability of the Pull-Moran applied on a clique as the underlying graph. Then we improve the result of the so called Bridge Thermal Theorem for the case of regular graphs in both the standard Moran and the Pull-Moran processes.

3.7.1 The Pull-Moran on Clique Graphs

First, we consider the PULL-Moran on a clique of size N. Note that considering the clique as the underlying graph can be interpreted as assuming the population to be homogeneous with no spatial structure [18, 64, 65]. Let $S = \{1, 2, \cdots, N\}$ be the set of states and let state $i \in S$ be the number of infected vertices. Particularly, $i = 0$ and $i = N$ are the extinction and fixation states respectively. We define the fixation probability Q_i as the probability of reaching fixation from state i. In other words, Q_i is the fixation probability given that i many infected nodes currently exist in the network. Moreover, according to the definition of the PULL-Moran, given a state i then the next state after one round of the process can only be $i+1$ or $i-1$ or i itself. In this section we will calculate Q_1, which is the fixation probability of the PULL-Moran protocol given a randomly placed mutant with fitness r, in a non-structured population.

We have

$$Q_i = P_{i,i-1}Q_{i-1} + (1 - P_{i,i-1} - P_{i,i+1})Q_i + P_{i,i+1}Q_{i+1} \tag{3.29}$$

where $P_{i,i+1}$ is the probability of moving from state i to $i+1$ and $P_{i,i-1}$ is the probability of moving from state i to $i-1$.

If $\mathbf{P} = (P_{i,j})$ is the transition matrix of the PULL-Moran model then in order to find a solution $Q = (Q_0, Q_1, \ldots, Q_n)$ we need to solve the equation system $\mathbf{P}Q = Q$ subject to the boundary conditions $Q_0 = 0$ and $Q_N = 1$. Solving this system of equations is not always easy, but here the symmetric shape of the clique helps to simplify the problem. Here, we follow the proof in [65]. From Eqn. (3.29) we get

$$P_{i,i-1}(Q_i - Q_{i-1}) = P_{i,i+1}(Q_{i+1} - Q_i).$$

Dividing both sides by $P_{i,i+1}$ we get

$$\frac{P_{i,i-1}}{P_{i,i+1}}(Q_i - Q_{i-1}) = Q_{i+1} - Q_i. \tag{3.30}$$

Defining $\gamma_i = \frac{P_{i,i-1}}{P_{i,i+1}}$ as the *birth-death rate* and letting $h_i = Q_i - Q_{i-1}$ in Eqn. (3.30), it follows that

$$h_{i+1} = \gamma_i h_i \tag{3.31}$$

Using recursion

$$h_{i+1} = h_1 \prod_{j=1}^{i} \gamma_j. \tag{3.32}$$

Also, based on the boundary conditions

$$\sum_{i=0}^{N-1} h_{i+1} = Q_N - Q_0 = 1 - 0 = 1 \tag{3.33}$$

We have

$$1 - h_1 = -h_1 + \sum_{i=0}^{N-1} h_{i+1} = \sum_{i=1}^{N-1} h_{i+1} = h_1 \sum_{i=1}^{N-1} \prod_{j=1}^{i} \gamma_j$$

and therefore from Eqn. (3.33)

$$h_1 + h_1 \sum_{i=1}^{N-1} \prod_{j=1}^{i} \gamma_j = 1. \tag{3.34}$$

Since $h_1 = Q_1 - Q_0 = Q_1$ then

$$Q_1 = \frac{1}{1 + \sum_{i=1}^{N-1} \prod_{j=1}^{i} \gamma_j}. \tag{3.35}$$

Now, we need to calculate $P_{i,i+1}$ and $P_{i,i-1}$. For a clique with N vertices we have

$$P_{i,i+1} = \frac{N-i}{N} \cdot \frac{ri}{ri + N - i - 1}.$$

In simple terms, after choosing a non-infected vertex (with probability $\frac{N-i}{N}$), we need to choose one of its infected neighbours (with probability $\frac{ri}{ri+N-i-1}$).

Similarly,

$$P_{i,i-1} = \frac{i}{N} \cdot \frac{N-i}{r(i-1) + N - i}.$$

Consequently,

$$\gamma_i = \frac{P_{i,i-1}}{P_{i,i+1}} = \frac{\frac{i}{N} \cdot \frac{N-i}{r(i-1)+N-i}}{\frac{N-i}{N} \cdot \frac{ri}{ri+N-i-1}} > \frac{1}{r}$$

for $r > 1$.

Finally, from Eqn. (3.35), it follows

$$Q_1 = \frac{1}{1+\sum_{i=1}^{N-1}\prod_{j=1}^{i}\gamma_j} < \frac{1}{1+\sum_{i=1}^{N-1}\frac{1}{r^i}} = \frac{1-\frac{1}{r}}{1-\frac{1}{r^n}}$$

where $\frac{1-\frac{1}{r}}{1-\frac{1}{r^n}}$ is the fixation probability of any vertex in a regular graph subject to the standard Moran process.

3.7.2 The Bridge Isothermal Theorem

In the next observation, by considering regular graphs as the population structure, we reduce the general upper bound on the fixation probability in the Bridge Thermal Theorem.

Observation 3.11. *Let $f_r(v)$ and $f'_r(v)$ be the fixation probabilities of vertex $v \in V$ applying the standard Moran and the* Pull-*Moran respectively on a regular graph $G(V, E)$. If $f_r(v) \geq 1-q$ for all vertices v and $r > 1$ then we have $f'_r(v) \leq r \cdot q$.*

Proof. Let d be the degree of every vertex in G. Similarly to Eqn. (3.19), assume that

$$f'_r(v) > r \cdot q. \tag{3.36}$$

Then, by replacing $\deg u = d$ for every v in Eqns. (3.21)-(3.25), similarly to Eqn. (3.26), it follows that

$$\begin{aligned} \frac{1}{r}\cdot\frac{rd}{rd+r+d-1} \geq \frac{1}{r}\cdot f'_r(v) > q \geq \frac{1}{r+1} \\ \Rightarrow \frac{rd}{rd+r+d-1} > \frac{1}{r+1} \\ \Rightarrow rd+d > rd+r+d-1 \\ \Rightarrow r < 1. \end{aligned} \tag{3.37}$$

which contradicts with the fact that $r > 1$. □

Although Observation 3.11 reduces the upper bound of the fixation probability for regular graphs by a factor of r in comparison to the Bridge Thermal Theorem 3.9, the upper bound is still large. Precisely, the original *Isothermal Theorem* for the standard Moran process in [18] indicates that for all regular graphs (graphs where all edges have the same degree), the fixation probability $f_r(G) = \frac{1-1/r}{1-1/r^n}$. Since G is regular we have

$$f_r(v) = f_r(G) = \frac{1 - 1/r}{1 - 1/r^n} \geq 1 - 1/r \tag{3.38}$$

and applying Theorem 3.11, $f'_r(v) \leq r.\frac{1}{r} = 1$ which is not helpful.

In the next theorem we reduce the upper bound introduced in Observation 3.11 subject to a more restricted value of r.

Theorem 3.12 (Bridge Isothermal Theorem)**.** *Let $f_r(v)$ and $f'_r(v)$ be the fixation probabilities of vertex $v \in V$ applying the standard Moran and the* Pull*-Moran respectively to a regular graph $G(V, E)$. Then $f'_r(G) \leq f_r(G) = \frac{1-1/r}{1-1/r^n}$ for $r \geq \sqrt{d}+1$ where d is the degree of each $v \in V$.*

Proof. We fix $r \geq \sqrt{d} + 1$. Let $f_r(v)$ and $f'_r(v)$ be the fixation probabilities of vertex $v \in V$ while applying the standard Moran and the Pull-Moran respectively on a graph $G(V, E)$. We prove our claim by contradiction. Suppose

$$f'_r(v) > \frac{1 - 1/r}{1 - 1/r^n} \tag{3.39}$$

for some $v \in V$.

Moreover from Eqn. (3.23) and Eqn. (3.39) we have

$$\frac{\sum_{u \in N(v)} \frac{r}{r+\deg u-1}}{1 + \sum_{u \in N(v)} \frac{r}{r+\deg u-1}} \geq f'_r(v) > f_r(v) = \frac{1 - 1/r}{1 - 1/r^n}. \tag{3.40}$$

Since $\deg v = \deg u = d$ and $\frac{1-1/r}{1-1/r^n} > 1 - \frac{1}{r}$

$$\begin{aligned} &\frac{\frac{d \cdot r}{r+d-1}}{1 + \frac{d \cdot r}{r+d-1}} > 1 - \frac{1}{r} \\ \Rightarrow &1 + \frac{r + d - 1}{d \cdot r} < \frac{r}{r - 1} \\ \Rightarrow &(r-1)^2 < d \\ \Rightarrow &r < \sqrt{d} + 1. \end{aligned} \tag{3.41}$$

which contradicts with our assumption. Thus,

$$f'_r(v) \leq \frac{1 - 1/r}{1 - 1/r^n} \tag{3.42}$$

for all $v \in V$ and therefore

$$f'_r(G) \leq f_r(G).$$

□

3.8 A Class of Selective Amplifiers for the Pull-Moran Process

In this section we introduce a class of graphs $\mathcal{G} = \{G_n : n \geq 1\}$ called *punk* graphs. Here $G_n(V, E)$ has $|V| = n \cdot k + n$ vertices consisting of a ring with $n \cdot k$ vertices and an independent set of n vertices in the ring. Each vertex in the independent set is connected to exactly k vertices, see Figure 3.5. Considering the fixation power of the center of a star graph, the punk graph is basically formed from merging several stars together. For ease of reference, a vertex in the ring is called a *ring vertex* and a vertex in the independent set is called a *blade vertex*. We want to show that the blade vertices are amplifiers. Note that punk graphs are formed out of merging n many stars with k many leaves together. Here we are inspired by the fact that an infected center of a star graph would amplify the fixation.

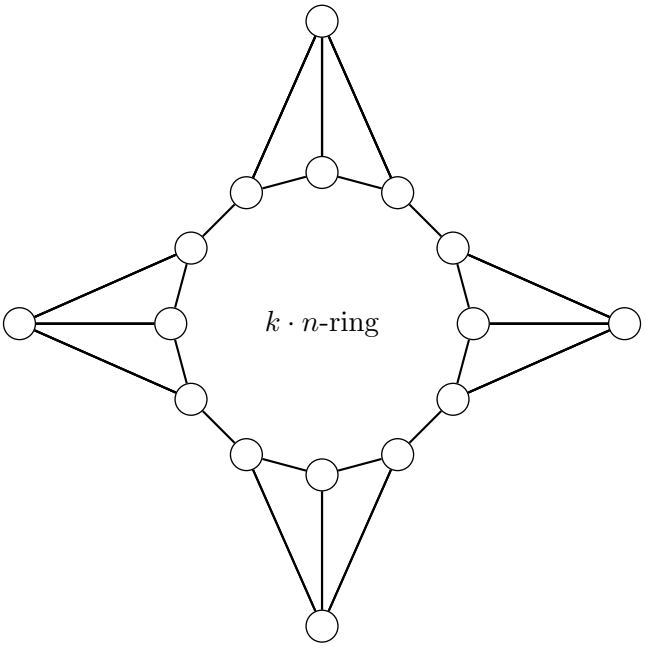

FIGURE 3.5: The punk graph with $n \cdot k + n$ vertices ($k = 3, n = 4$)

Finding a lower bound on the fixation probability of the punk graph with an infected blade vertex is not as easy as for other symmetric graphs. Unlike star or urchin graphs, which only have two different stochastic processes, in punk graphs all of the ring vertices connected to a blade vertex have different stochastic processes. This makes it difficult to

prove that punk graphs are selective amplifiers theoretically. However, our simulations for different numbers of n and r shows that the rate of fixation, given an infected blade vertex, is relatively high. Figure 3.6 shows the fixation rate of the aforementioned process, where $n = k$ and $r = 3, 5$ and 10. Here we can observe that for a large enough value of r (≥ 5), the fixation is independent of r as n (or k) grows. Moreover, for large values of n the fixation is almost inevitable.

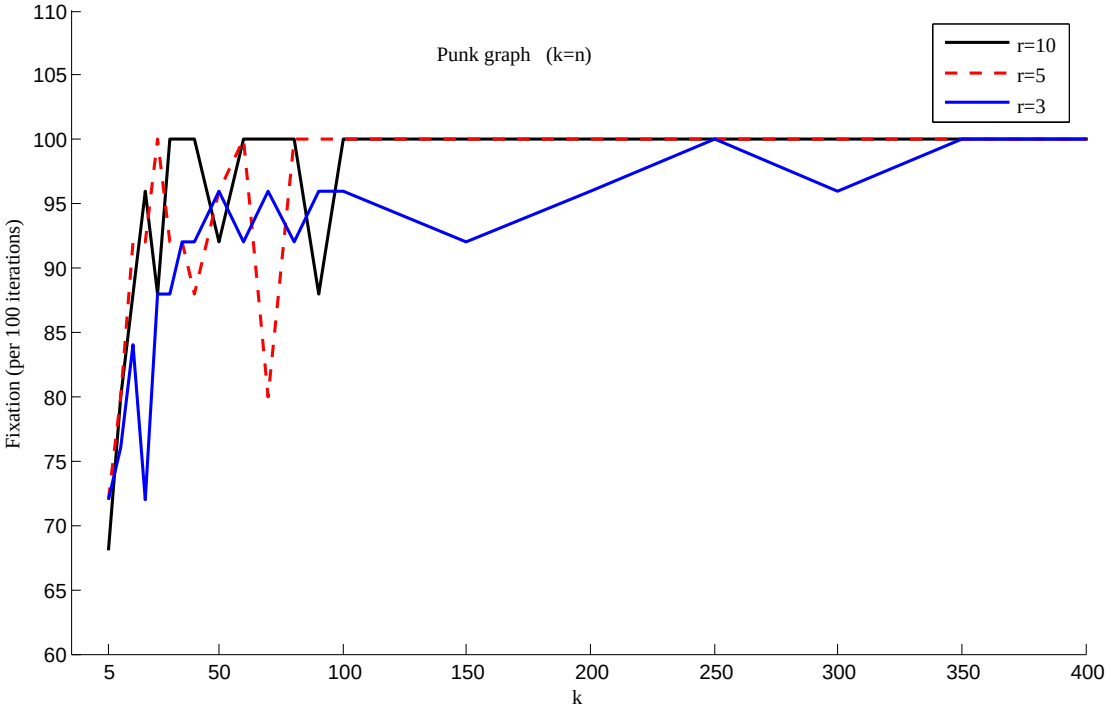

FIGURE 3.6: Fixation rate of the PULL-Moran process given an infected blade

Now, in order to find a practical lower bound on the total fixation probability of the punk graph, we will carry out a more detailed analysis. Let V_1 (V_2) be the set of all ring vertices (all blade vertices) and for each $v \in V_1$ ($u \in V_2$), let $f_r(v)$ ($f_r(u)$) be its corresponding fixation probability. As mentioned previously, the diversity of ring vertices in terms of different stochastic processes is our main challenge. Therefore, we define $f_1 = \min_{v \in V_1} f_r(v)$ as the lower bound for all the fixation probabilities of ring vertices. Also let $f_2 = f_r(u)$ for each $u \in V_2$. Note that the fixation probabilities of all blade vertices are equal. Now, according to the definition of the total fixation probability, we have

$$f_r(G) \geq \frac{n \cdot k}{n \cdot k + n} f_1 + \frac{n}{n \cdot k + n} f_2. \tag{3.43}$$

First, we estimate f_1. There exists a ring vertex $v \in V_1$, with $f_r(v) = f_1$. Assume that v is connected to ring vertices w_1, w_2 and a blade vertex u. Let $\{v, u\}$ be the state where vertices v, u are both infected. Similarly, let $\{v, w_1\}$ and $\{v, w_2\}$ be the states where

v, w_1 and v, w_1 are infected, respectively. Also let $\varnothing$ be the extinction state. Now, we have

$$\begin{aligned} f_1 = p_1 \cdot f_r(\varnothing) + p_2 \cdot f_r(\{v,u\}) + \frac{p_3}{2} \cdot f_r(\{v,w_1\}) \\ + \frac{p_3}{2} \cdot f_r(\{v,w_2\}) + (1 - p_1 - p_2 - p_3) \cdot f_1, \end{aligned} \tag{3.44}$$

where p_1 is the probability that v becomes uninfected, p_2 is the probability that v infects u and finally p_3 is the probability that v infects either w_1 or w_2.

Since $f_r(\varnothing) = 0$, $f_r(\{v,u\}) > f_2$, $f_r(\{v,w_1\}) > f_1$ and $f_r(\{v,w_2\}) > f_1$, then from Eqn. (3.44) we have

$$f_1 > p_2 \cdot f_2 + p_3 \cdot f_1 + (1 - p_1 - p_2 - p_3) \cdot f_1.$$

Therefore,

$$f_1 > \frac{p_2}{p_1 + p_2} \cdot f_2 = \frac{\frac{r}{r+k-1}}{1 + \frac{r}{r+k-1}} \cdot f_2$$

since $p_2 = \frac{1}{|V|} \cdot \frac{r}{r+k-1}$ and $p_1 = \frac{1}{|V|} \cdot 1$.

Hence,

$$f_1 > \frac{r}{2r + k - 1} \cdot f_2 \tag{3.45}$$

Finally, from Eqn. (3.43) and Eqn. (3.45), it follows

$$f_r(G) \geq \frac{k(r+1) + 2r - 1}{(2r + k - 1)(k+1)} \cdot f_2. \tag{3.46}$$

The next set of simulations are based on the above analysis. Here, for different values of k ($k = n$), we simulate the LHS of equation Eqn. (3.46) by randomly placing the first mutant in the punk graph and counting the number of fixations in 100 iterations. We also calculate the RHS of Eqn. (3.46) by simulating f_2, that is the rate of the fixation given an infected blade vertex, and then multiply it by $\alpha(k,r) = \frac{k(r+1)+2r-1}{(2r+k-1)(k+1)}$, see Figures 3.7, 3.8 and 3.9.

In these three figures, the *red*, *black* and *blue* lines show $\alpha(k,r) \cdot f_2$, $f_r(G)$ and f_2 respectively. All three figures show that the lower bound that is calculated in Eqn. (3.46)

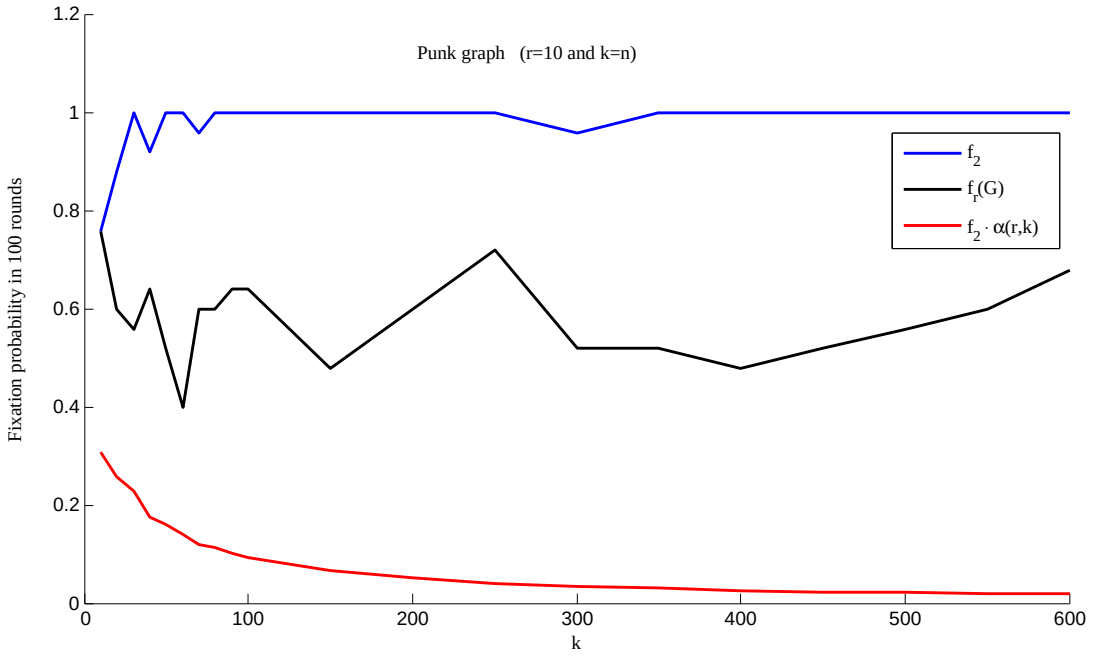

FIGURE 3.7: Fixation rate of $f_r(G)$, f_2 and $\alpha(k,r) \cdot f_2$ with $n = k$ and $r = 10$

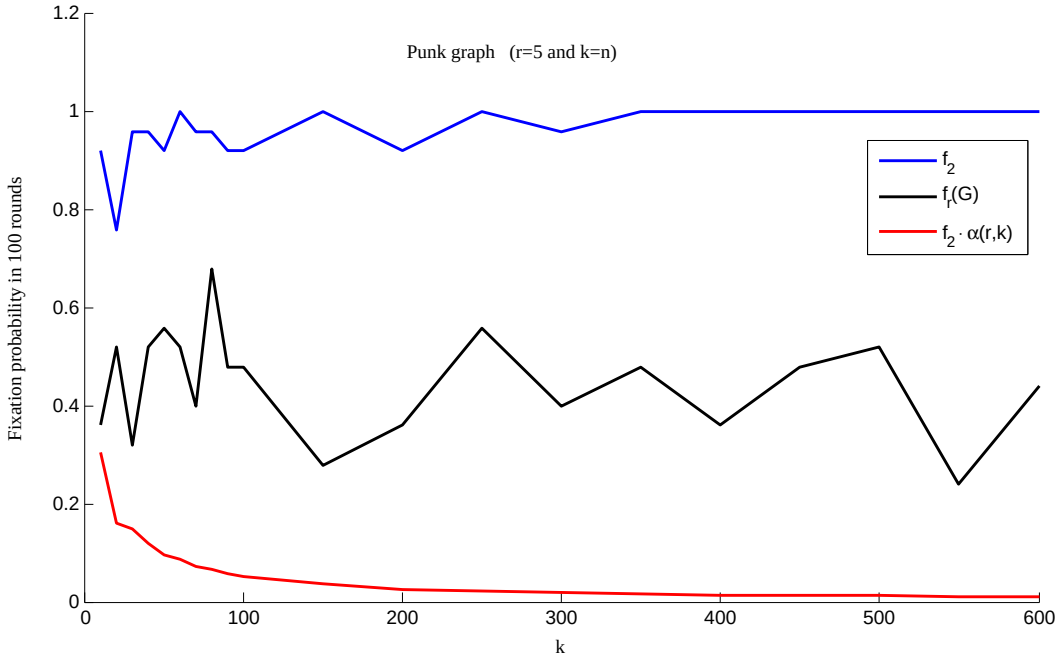

FIGURE 3.8: Fixation rate of $f_r(G)$, f_2 and $\alpha(k,r) \cdot f_2$ with $n = k$ and $r = 5$

is valid. In particular, for smaller values of r, our simulations show that the approximated lower bound (the red line) is closer to the actual value of the total fixation probability (the black line). However, generally speaking, the approximated lower bound shrinks as n (or k) grows. This is not unexpected because $f_2 \leq 1$ and $\alpha(k,r)$ decreases as k increases. So we can conclude that our analysis of the punk graph leads to a valid lower

bound for the total fixation probability of the punk graph. However, the approximate bound for the fixation probability of the ring vertices (*i.e.*f_1) is not possibly good enough to give us a better result.

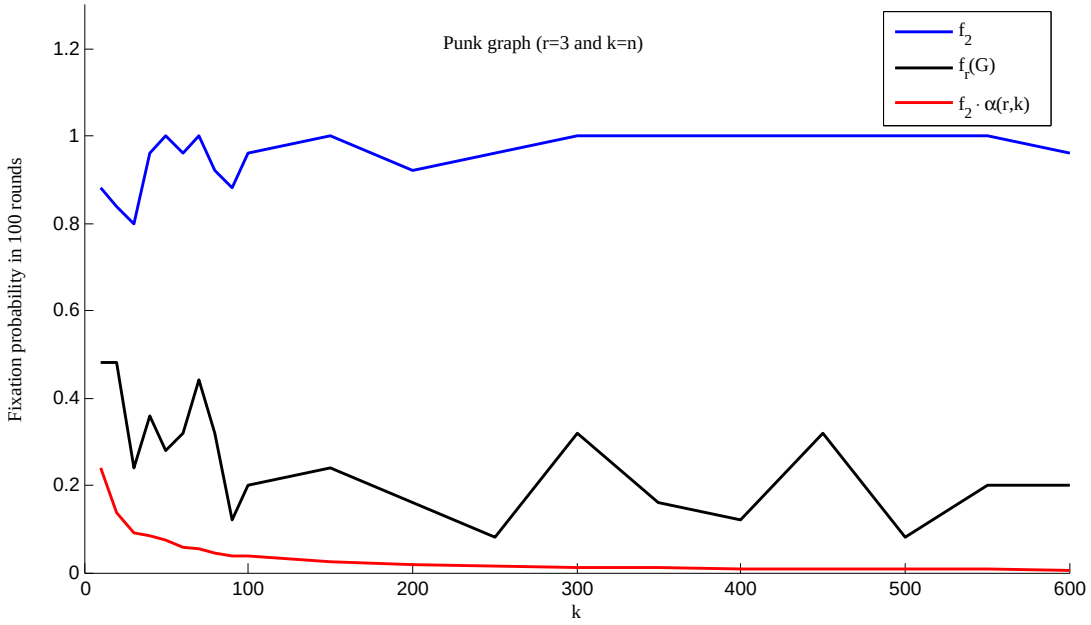

FIGURE 3.9: Fixation rate of $f_r(G)$, f_2 and $\alpha(k,r) \cdot f_2$ with $n = k$ and $r = 3$

3.9 Conclusion

In this chapter we analysed an evolutionary based process called the PULL-Moran process (also known as death-birth update), motivated by studies in [18, 19]. The advantage of the PULL-Moran process is that it is independent of any global information such as the total number of infected nodes in a network. We proved the existence of *strong* selective suppressors. Following the definition of strong universal amplifiers in [19], we showed that a class of strong universal amplifiers does not exist for the PULL-Moran process. We showed that any class of selective amplifiers in the (standard) Moran process is a class of selective suppressors under the PULL-Moran process. As for regular graphs, we proved that the total fixation probability of a regular graph in the PULL-Moran process is upper bounded by the total fixation probability of the same graph under the standard Moran process. Finally, we introduced a class of graphs called punk graphs. Through simulations, we showed that they are good candidates for amplifying the fixation process under the PULL-Moran protocol. However, calculating a promising theoretical lower bound on the total fixation probability of punk graphs will remain an open question.

Chapter 4

Coalescing-Branching Walk

4.1 Introduction

In this section we will analyse a type of random walk called the *coalescing-branching* (COBRA) random walk. COBRA walks, first introduced in [4], are a modified version of the standard random walk, with the main parameter called the *branching factor* k. Like the simple random walk, COBRA walks start from an arbitrary vertex that is labelled *active*. In practice, any active node may possess a piece of information (a rumour) or be infected by a virus. We are interested in the propagation of this rumour or virus throughout the network. A COBRA walk is an iterative process where at each iteration every active node chooses k many neighbours *u.a.r*, which are then activated in the next iteration. This is called the *branching* feature, where each active vertex initiates k independent random walks. The other distinguishing feature of the COBRA walk is its *coalescence*. Here, all the independent random walks that choose the same vertex in an iteration will coalesce into one single walk. More precisely, a vertex is active at a time step t if and only if it is chosen by an active node at time step $t-1$. The COBRA walk can be analysed as a modification of known processes such as simple/multiple random walks [66, 67]. It can also be observed in a *rumour spreading* setting [68, 69]. In the following, we review some of the studies carried out in these frameworks and show the similarities and differences between the COBRA walk and other processes.

4.1.1 Random Walks

A simple *random walk* is a stochastic process (a finite Markov chain) on a given graph with an arbitrary starting point. During each iteration, a random walk moves from the current node to one of the neighbouring nodes which is chosen *u.a.r*. Note that a

simple random walk is basically a cobra walk with branching factor 1. One of the most important parameters when analysing the efficiency of a random walk is the *cover time*. The cover time of a random walk on an underlying graph (or network) is the expected time taken for a random walk to visit all of the vertices at least once [70, 71]. Bounding the cover time of random walks on different graphs has been extensively studied [72–75]. Particularly, in [76, 77], Feige calculates general upper and lower bounds for the cover time. He shows that the cover time of a random walk on all undirected connected graphs of size n takes a value between $\Theta(n \log n)$ and $\Theta(n^3)$. He also shows that the bounds are tight *i.e.* $\Theta(n \log n)$ is the cover time for clique graphs and $\Theta(n^3)$ is the cover time for the graph called the lollipop.

4.1.2 Parallel Random Walks

The parallel random walks (also known as many random walks) approach is one of the similar known approaches to the COBRA walk. A special case of the parallel walks process is introduced in [78] where the starting points are picked out of the stationary distribution. Cooper, Frieze and Radzik in [66] investigate different random walk models such as multiple walks, particles with a finite life, talkative particles, predator-prey, sticky particles and explosive particles. In [67], Alon et al. study the expected cover time of k-many independent random walks on different graph classes such as cycle, d-dimensional grid, hypercube and expander, as summarised in Table 4.1. Particularly,

TABLE 4.1: Result summary of parallel random walks in [67] ($\forall \varepsilon > 0$)

Graph	Cover time	sped-up cover time
Cycle	$n^2/2$	$\Theta\left(\frac{n^2/2}{\log k}\right)$
2-dimensional grid	$\Theta(n \log^2 n)$	$\Theta\left(\frac{n \log^2 n}{k}\right)$, $k < O(\log^{1-\varepsilon})$
d-dimensional grid, $d > 2$	$\Theta(n \log n)$	$\Theta\left(\frac{n \log n}{k}\right)$, $k < O(\log^{1-\varepsilon})$
Hypercube	$\Theta(n \log n)$	$\Theta\left(\frac{n \log n}{k}\right)$, $k < O(\log^{1-\varepsilon})$
Complete graph	$\Theta(n \log n)$	$\Theta\left(\frac{n \log n}{k}\right)$, $k < n$
Expanders	$\Theta(n \log n)$	$\Theta\left(\frac{n \log n}{k}\right)$, $k < n$

for the special case of expander graphs they prove a linear speed-up when $k \leq n$. They also show that if there is a large gap between the cover time and the hitting time of a graph then k random walks (for k sufficiently small) covers the graph k times faster than a single random walk. Here, the *hitting time* of a graph is the expected time taken for a random walk to move from a vertex v to a vertex u maximized over all pairs of u and v. In [79], applying k independent random walks, the authors calculate the precise upper and lower bounds on the order of speed-up in different graph classes. Comparing COBRA walk and k parallel random walks, the two protocols are in some ways similar since, in every iteration, multiple nodes may select their neighbours *u.a.r.* However, there are some differences between the two mechanisms. As an example, the expected cover time of k parallel random walks on a line graph is $\Omega(n^2/\log k)$ for $1 \leq k \leq n$ (see [67]) whilst a COBRA walk, with branching factor $k = 2$, has an expected cover time of $O(n)$ on the same graph. Furthermore, unlike the COBRA walk, where the number of active nodes may be changing at every iteration, in k parallel random walks the number of active nodes is always constant and is equal to k. Moreover in the COBRA walk the process dependencies, caused by the coalescing mechanism, make it more challenging to analyse the cover time when compared to parallel random walks.

4.1.3 Rumour Spreading

Gossip-based (also known as *epidemic*) algorithms have been extensively applied for designing simple and powerful tools for propagating information within different networks. In this context, *randomised rumour spreading* is arguably the most well-studied approach in the class of gossip-based algorithms for disseminating information on a large scale. In this respect, PULL, PUSH and PUSH-PULL, introduced in [80], are widely studied methods in spreading rumours across a network. The main idea discussed in [80] is to design simple and reliable algorithms in order to propagate updates amongst all replica sites in a network. In all three of the aforementioned algorithms, the rumour is initially received by a vertex chosen *u.a.r* and then spread among all the vertices iteratively. In the PUSH protocol, at each iteration every *active* node that has received the rumour in a previous round then chooses a neighbour *u.a.r* and sends the rumour to that neighbour. In the PULL approach, at each iteration every inactive node chooses a neighbour *u.a.r* and then receives the rumour, if the chosen neighbour has the rumour. Finally, in the PUSH-PULL protocol, which is a combination of PUSH and PULL algorithms, in every iteration each node contacts a random neighbour then, if one of them has the rumour, it passes it on to the other one. Each of these approaches have been widely studied. As for the PUSH method, Feige, Peleg, Raghavan and Upfal, in [81], find a bound of $O(\log n)$ on the *broadcast time* (also known as cover time) of hypercube

and dense random graphs. In [82], Elsässer and Sauerwald show that the PUSH algorithm spreads a rumour in a n-dimensional star graph (different from the star graph) within $O(\log n)$ time. The same authors in [83] find a correlation between the mixing time and the broadcasting time, where a rapid mixing time leads to a fast broadcasting time. They also carried out some analysis on the broadcasting time of a certain class of Cayley graphs which includes star graphs, pancake graphs and transposition graphs. The broadcast time of the PUSH method on random graphs is studied in [84] where the authors show that the bound on the broadcast time in random regular graphs is the same as that of complete graphs. Later, Fountoulakis and Panagiotou generalise the bound in [84] to random regular graphs in [85]. Elsässer and Sauerwald, in [86] and Elsässer, in [87], study the broadcast time and the total number of the transmissions of the rumour of the PUSH-PULL strategy for random graphs. Another study is carried out in [56] on the broadcasting time of the PUSH-PULL method in random regular networks. Karp, Schindelhauer, Shenker and Vöcking study the PUSH-PULL strategy for complete graphs in [88]. Specifically, they show that the PUSH-PULL strategy reduces the number of transmissions of the rumour in the PUSH protocol, that is $\Theta(n \log n)$, down to $\Theta(n \log \log n)$.

One of the most studied methods in measuring the broadcast time of randomised rumour spreading in a network is to investigate the expansion properties of the network. Specifically, the *conductance* (see Definition 4.6) of a graph has been widely studied as a measure of the graph expansion. The conductance of a (connected) graph (also known as the Cheeger constant), denoted by ϕ, is the minimum ratio of the edges leaving a set of vertices over the edges incident to that set [68]. In other words, in well connected graphs the conductance is large, and it is small for graphs that are not well connected. In this respect, Chierichetti, Lattanzi and Panconesi [11] show that the PUSH and the PULL strategies are not necessarily very fast in networks that have a large conductance. In particular they prove that the expected broadcast time of the aforementioned strategies on the star graph, with a constant conductance, is of polynomial order. However, using the PUSH-PULL strategy, the same authors in [89] show that the broadcast time for any graph is $O(\phi^{-6} \log^4 n)$ with high probability. The authors in [69] improve the bound in [89] to $O((\log \phi^{-1})^2 \phi^{-1} \log n)$. Finally, Giakkoupis in [68] shows that for any graph with n vertices and conductance ϕ, and for any start vertex, the broadcast time of the PUSH-PULL strategy is $O(\phi^{-1} \log n)$ rounds with high probability. He also shows that the aforementioned bound is tight for the case where $\phi = \Omega(1/n)$. Among all the reviewed methods, the rumour spreading PUSH protocol is possibly the closest approach to the COBRA walk. Recall that at each iteration in the PUSH protocol every active node chooses a neighbour *u.a.r* and then pushes the information to the neighbour. Again, comparing the push based rumour spreading strategy and the COBRA walk, they

are somewhat similar as, in every iteration, several nodes can be active in a given step. However, the coalescing mechanism introduces a significant difference between the COBRA walk and the PUSH protocol. More precisely, the number of active nodes in rumour spreading is monotonically non-decreasing whereas in a COBRA walk the number of active nodes may decrease due to the coalescing mechanism. However, by introducing two new protocols namely COBRA PUSH and COBRA PULL (see Section 4.4), we are able to analyse the COBRA protocol as a rumour spreading approach.

4.1.4 Cobra Walks

Apart from biological aspects, the network structure of an affected population is of great significance in the study of epidemic diseases. In order to find out how likely a disease can spread among the individuals in a population, we need to carefully observe the underlying social network of that population. In computer science, the SIS (Susceptible Infected Susceptible) model attempts to capture the dynamics in the spreading of information, for example a rumour or a computer virus, in a fixed population, for example social networks or social networks. In SIS, there is no restriction on the possibility of reinfection. Conversely, in the SIR (Susceptible Infected Recovered) model, re-infection is not allowed. SIS-type epidemic models have been extensively studied [10, 90, 91]. Here, the main focus is on the persistence time and the epidemic density of such models. In [4], Dutta, Pandurangan, Rajaraman and Roche introduce the COBRA walk in order to investigate the time taken for a SIS-type process to affect a large fraction or the entirety, of a network, *i.e.* the cover time. The cover time of the COBRA walk has been observed on different graph classes in [4]. Here, the authors show that the cover time of the COBRA walk for tree, finite 2-dimensional grids and finite d-dimensional grids are $O(n \log n)$, $O(\sqrt{n} \log n)$ and $O(n^{1/d} \log n)$ respectively. For a complete graph they show a cover time of $O(\log n)$. Particularly, their main result is a high probability bound, of size $O(\log^2 n)$, for the cover time of COBRA walks on d-regular expanders subject to a sufficiently large k. To achieve that, they consider two different phases. In the first phase they show that there are at least δn ($\delta > 0$) number of vertices covered after $O(\log n)$ time steps. In the second phase, after the COBRA walk activates a set of size $\Omega(n)$, a different approach is used to prove that the COBRA walk can achieve a full coverage in $O(\log^2 n)$ time. In simple random walks, in order to calculate the maximum hitting time, one could easily run the algorithm for $O(\log n)$ periods of length equal to the maximum hitting time and the result follows from Matthews' Theorem 4.3. However, as for COBRA walks, this is not very straightforward as we know nothing about the distribution of the δn activated vertices. In the following we will show that it is possible

to approach the COBRA walk protocol using a different method in order to improve the broadcasting time on d-regular expanders.

4.2 Our Contribution

Improving the broadcasting time of the COBRA walk analysed in [4], for regular expander graphs, is the main motivation behind this chapter. In this chapter we analyse the broadcasting time of the COBRA protocol for *random regular* graphs. Here, we look into the COBRA approach as a randomised rumour spreading protocol, see Section 4.1.3. In order to take advantage of rumour spreading features, we consider a COBRA walk as a modified PUSH strategy called the COBRA PUSH protocol. We then show that it is possible to improve the result in [4] for random regular graphs. In the following, we will first define COBRA PUSH, COBRA PULL and COBRA PUSH & PULL protocols. Then, in Section 4.5, we will show that $T = O(\log n)$ rounds of COBRA PUSH & PULL is enough to broadcast a rumour to all vertices of a random regular graph. Finally we will show that the same amount of time, *i.e.* $T = O(\log n)$, is only sufficient for the COBRA PUSH protocol to cover the graph.

4.3 Model and Definitions

All the definitions of this section are mostly extracted from [4]. Let $G(V, E)$ be a connected undirected graph where V and E are the vertex and edge set respectively. According to [4], a *coalescing-branching* (COBRA) random walk on G with branching factor k starts at some arbitrary $v \in V$. In the initial state at $t = 0$ a pebble is placed at v. Then in every iteration, every pebble in G copies itself $k-1$ times. This means that at the end of the copying phase k many pebbles exist at the vertices which originally had a pebble. Thereafter, each pebble independently chooses a neighbour of its current vertex *u.a.r* and moves to it. This is called branching. After the new placement has occurred, if two or more pebbles choose the same vertex they coalesce into a single pebble, and the process begins again. The process is summarised in Algorithm 4.3.1. Note that in a COBRA walk, a vertex may be visited more than once. An active set, S_t, is the set of all vertices of G that have a pebble at time t.

As [4] is the main motivation behind our study in this section, we will show some of their key definitions and results.

Definition 4.1. *The inclusive neighbourhood of S denoted by $N(S)$ is*

$$S \cup \{v : (v, w) \in E \text{ for some } w \in S\},$$

Algorithm 4.3.1 The COBRA random walk with branching factor k

while all vertices are visited by a COBRA-walk (pebble) **do**
  for all pebble **do**
    Clone the pebble, located at vertex v, $k-1$ times
    Choose one of the neighbours (of v) uniformly at random and move the pebble to it
    Coalesce the pebbles arrived on the same vertex

and $\partial S = N(S) \setminus S$ *is defined as the* outer boundary *of* S.

Definition 4.2 ([4]). *The cover time of a* COBRA *walk on* G *is defined by* $\max_{v \in V} \tau_v$ *where* τ_v *is the minimum time for a* COBRA *walk, starting from* v*, such that* $u \in S_t$ *for all* $u \in V - v$ *for some* $t \leq \tau_v$*. Similarly, the expected cover time of a* COBRA *walk is defined by* $\max_{v \in V} \mathbb{E}[\tau_v]$.

4.3.1 General Bounds and Matthews' Theorem

Several studies have been carried out on identifying general bounds for the cover time of arbitrary graphs. Aleliunas, Karp, Lipton, Lovász and Rackoff [92] show that for any graph $G(V, E)$ the cover time $C(G)$ has an upper bound of $O(2|E|(|V|-1))$. Feige introduces general upper [93] and lower [94] bounds on the cover time of a random walk on an arbitrary connected graph. More precisely, he shows that

$$(1 - o(1)) \cdot n \log n \leq C(G) \leq (1 + o(1)) \cdot \frac{4}{27} \cdot n^3$$

and that both bounds are tight. The expected cover time of a complete graph is $\Theta(n \log n)$ and the expected cover time of a *lollipop* graph (a path of length $n/3$ connected to a clique of size $2n/3$) is $\Theta(n^3)$ respectively. Another important result that shows a very tight connection between the cover time and hitting time of a connected graph is introduced by P. Matthews in [95, 96].

Theorem 4.3 (Matthews' Theorem [95, 96]). *Let* $C(G)$ *be the cover time from any node of a connected graph*

$$h_{min} \cdot H_n \leq C(G) \leq h_{max} \cdot H_n$$

where h_{min} *(*h_{max}*) is the minimum hitting time (maximum hitting time) and*

$$H_k = 1 + 1/2 + \cdots + 1/n = \ln(k) + \Theta(1)$$

is the k*th harmonic number.*

A simple but intuitive proof of the Matthews' theorem for a weaker upper bound of $2\log_2 n$, discussed in [38], is as follows.

Lemma 4.4 (Lemma 2.8. [38])**.** *Let b be the expected number of steps before a random walk visits more than half of the nodes, and let h be the maximum hitting time between any two nodes. Then $b \leq 2h$.*

Proof. Without loss of generality assume that $n = 2k + 1$ is an odd number. Let t_v be the first time that vertex v is visited. Also let C_h be the time it takes for the random walk to visit more than half of the vertices *i.e.* $k + 1$ vertices. Therefore

$$\sum_v t_v \geq (k+1)C_h$$

Hence

$$\begin{aligned} b = \mathbb{E}[C_h] &\leq \frac{1}{k+1}\sum_v \mathbb{E}[t_v] \\ &\leq \frac{n}{k+1}h < 2h \end{aligned} \tag{4.1}$$

□

According to Lemma 4.4, in $2h$ steps the random walk visits more than half of the vertices. Following the same argument, in the next $2h$ steps more than half of the remaining unvisited vertices will be covered. Therefore after about $2h\log_2 n$ steps all the vertices will be visited.

4.3.2 Expander Graphs and Measures of Expansion

Expander graphs are connected undirected graphs with two very distinctive properties; they are relatively sparse, *i.e.* they have relatively few number of edges, and they have strong connectivity properties. The extent of connectivity of expander graphs is measured by quantities such as the *vertex expansion* or *edge expansion*. Let $G(V, E)$ be an undirected and connected graph. For every set of vertices $S \subseteq V$, let ∂S be the *outer boundary* of S, see Definition 4.1.

Definition 4.5. *The vertex expansion of G (also known as Cheeger constant), $0 < \alpha(G) \leq 1$ is defined by*

$$\alpha(G) = \min_{0<|S|\leq \frac{n}{2}} \frac{|\partial S|}{|S|}. \tag{4.2}$$

Intuitively, in well-connected expanders subject to the vertex expansion every set of vertices, that is not very large, should have many neighbours.

For every two sets of vertices $U, W \subseteq V$, let $E(U,W) = \{(u,w) \in E : u \in U, w \in W\}$ be the set of the edges between U and W. Also define the *volume* of $S \subseteq V$ as $vol(S) = \sum_{v \in S} d(v)$.

Definition 4.6. *The edge expansion of G (also known as conductance) $0 < \phi(G) \leq 1$ is defined by*

$$\phi(G) = \min_{\substack{S \subseteq V \\ 0 < vol(S) \leq vol(V)/2}} \frac{E(S, V \backslash S)}{vol(S)}. \tag{4.3}$$

Note that $vol(V) = 2|E|$. In [97], Sauerwald and Stauffer show that

$$(\delta/\Delta)\phi(G) \leq \alpha(G) \leq \Delta \cdot \phi(G), \tag{4.4}$$

where δ and Δ are the minimum and maximum degrees of G respectively. Intuitively, $\phi(G)$ is large for well-connected graphs and small for the graphs that are not well-connected. We will use Definition 4.2 and Definition 4.6 later in the proof of Lemma 4.10.

4.3.3 Random Regular Graphs

A random d-regular graph is a graph that is chosen *u.a.r* from the probability space of all d-regular graphs with the same number of vertices.

Definition 4.7 ([98])**.** *Suppose n and d such that $3 \leq d < n$ and $r \cdot n$ is even. Let $\mathcal{G}(n, dreg)$ be the probability space on all d-regular graphs where each graph has the same probability. Then we call an element of this probability space, a random d-regular graph.*

Several models for generating random regular graphs have been introduced [99–101]. The *pairing* model is one of the most studied models in this respect [100]. Consider a family of n sets of filled with d nodes ($n \cdot d$ nodes in total). Each of these sets is regarded as d-many copies of every vertex. Consider a random matching of these sets. The result is either a random d-regular graph or a graph with multiple edges or loops. In the latter, we do the algorithm again. Note that each random d-regular graph is an expander *w.h.p.* [102].

4.4 Cobra Push and Cobra Pull Protocols

Here, we look into the COBRA approach as a randomised rumour spreading protocol. In order to take advantage of rumour spreading features, we consider the COBRA walk

protocol as a new Push strategy called Cobra Push. Formally, in each round of the Cobra Push protocol every active node forwards the message to k randomly selected neighbours, see Algorithm 4.4.1.

Algorithm 4.4.1 The Push Cobra protocol

```
1: for every active node v do
2:    for each of k messages do
3:       selects a random neighbour of v and moves to it
```

We define nodes to be *active* if and only if they have received the message in the previous round (for the first time or more often) from one or more neighbours. In each round of the Cobra Pull protocol, every node sends requests to k randomly chosen neighbours and they, in turn, reply with the message if and only if they themselves are active, see Algorithm 4.4.2.

Algorithm 4.4.2 The Pull Cobra protocol

```
1: for every node v do
2:    sends requests to k randomly selected neighbours
3:    for each active selected neighbour do
4:       sends the rumour to v
```

The main result of this chapter is the following theorem.

Theorem 4.8. *The* Cobra *protocol covers random regular graphs in time* $O(\log n)$, w.h.p.

In the following, we show that $T = O(\log n)$ rounds of Cobra Push & Pull (see Algorithm 4.5.1) is enough to broadcast a rumour to all vertices of a random regular graph and finally we show that the same amount of time, *i.e.* $T = O(\log n)$, is enough for only the Cobra Push protocol (or the generic Cobra protocol) to cover the graph.

4.5 Cobra Push & Pull Protocol

The main result of this section is Theorem 4.9.

Theorem 4.9. *Let G be a random d-regular graph. The* Push *&* Pull Cobra *protocol covers G in time* $O(\log n)$, w.h.p.

Where, for $T = \Theta(\log n)$, the Push & Pull Cobra protocol is as follows, see Algorithm 4.5.1.

To prove Theorem 4.9, first we show that, after $\Theta(\log n)$ rounds of Cobra Push, more than half of the nodes are active.

Algorithm 4.5.1 The PUSH & PULL COBRA protocol

1: **for** T rounds **do**
2: COBRA PUSH
3: **for** T rounds **do**
4: COBRA PULL

Lemma 4.10. *Let G be a random d-regular graph, $d \geq 3$. Let $k \geq d$. There exists a constant $\beta > 1/2$ such that for some $T = \Theta(\log n)$ there are βn many active nodes in round T of* COBRA PUSH*,* w.h.p..

Proof. Let S_t be the set of active vertices in step t, $s_t = |S_t|$. Let $N(S_t)$ denote the inclusive neighbourhood of S_t, and let $n_t = |N(S_t)|$. Furthermore, let $\ell_t = n_t/s_t$ (then $|N(S_t)| = \ell_t \cdot |S_t|$, and in particular $|N(S_t)| - |S_t| = n_t - s_t = s_t \cdot \ell_t - s_t = (\ell_t - 1) \cdot s_t$).

The set-up and the first few steps of this proof (up until Eqn. (4.9)) follow that of the proof of Lemma 12 in [4]. Here, we want to show that for any $t \geq 0$, the cobra walk with active set S_t such that $s_t \leq \beta n$, $\mathbb{E}[s_{t+1}] \geq (1+\mu)s_t$ for some $\mu > 0$. Instead, we show that the number of nodes in $N(S_t)$ that are not selected by the COBRA walk is sufficiently small, *i.e.* $\mathbb{E}[|N(S_t) - S_{t+1}|] \leq n_t - (1-\mu)s_t$. For each $u \in N(S_t)$, we define a random variable X_u that takes value 1 if $u \notin S_{t+1}$ and 0 otherwise. Then $Pr[X_u = 1] = (1 - 1/d)^{k \cdot d_u}$, where d_u is the number of the neighbours of u in S_t. Therefore,

$$\mathbb{E}[|N(S_t) - S_{t+1}|] = \sum_{u \in N(S_t)} \mathbb{E}[X_u] = \sum_{u \in N(S_t)} (1 - 1/d)^{k \cdot d_u} \leq \sum_{u \in N(S_t)} \exp(-\frac{k \cdot d_u}{d}).$$

Since $\sum_{u \in N(S_t)} d_u = d|S_t|$ and the RHS of the above inequality is a convex function, $\sum \exp(-\frac{k \cdot d_u}{d})$ is maximised when all the values of d_u are equal to either 1 or d (with an exception of possibly one d_u). Now, let R_1 (respectively R_2) be the number of nodes in $N(S_t)$ with $d_u = 1$ ($d_u = d$). Then

$$R_1 + R_2 = |N(S_t)| \tag{4.5}$$

$$R_1 + dR_2 = d|S_t|, \tag{4.6}$$

solving for R_1 and R_2, we have

$$R_1 = \frac{d}{d-1}(|N(S_t)| - |s_t|) \tag{4.7}$$

$$R_2 = \frac{1}{d-1}(d|S_t| - |N(S_t)|). \tag{4.8}$$

Thus

$$\mathbb{E}[|N(S_t) - |S_{t+1}|] \leq \frac{d}{d-1}(|N(S_t)| - |S_t|) \cdot \exp(-k/d) + \frac{1}{d-1}(d \cdot |S_t| - |N(S_t)|) \cdot \exp(-k) \tag{4.9}$$

In our notation, the RHS becomes

$$\frac{d}{d-1}(\ell_t - 1) \cdot s_t \cdot \exp(-k/d) + \frac{1}{d-1}(d - \ell_t) \cdot s_t \cdot \exp(-k).$$

We want to show that

$$\ell_t s_t - \left[\frac{d}{d-1}(\ell_t - 1) \cdot s_t \cdot \exp(-k/d) + \frac{1}{d-1}(d - \ell_t) \cdot s_t \cdot \exp(-k)\right] \geq (1+\mu)s_t \tag{4.10}$$

for some constant $\mu \in (0, 1)$. $\ell_t s_t$ is the maximum number of active nodes we can hope for in the next step. (Notice that ℓ_t is a static property of the graph, depending on the expansion – for any subset of vertices there is a fixed neighbourhood, and this ℓ_t simply expresses the size of that for whatever happens to be our current S_t.) From that we subtract the "bad nodes", those counted inside the big square brackets. The result should at least be larger by a constant factor > 1 than what we have got now, in step t.

We can divide through by s_t and obtain

$$\ell_t - \left[\frac{d}{d-1}(\ell_t - 1) \cdot \exp(-k/d) + \frac{1}{d-1}(d - \ell_t) \cdot \exp(-k)\right] \geq 1 + \mu. \tag{4.11}$$

We solve this for μ:

$$\begin{aligned}
\mu &\leq \ell_t - \left[\frac{d}{d-1}(\ell_t - 1)\cdot\exp(-k/d) + \frac{1}{d-1}(d-\ell_t)\cdot\exp(-k)\right] - 1 \\
&= (\ell_t - 1) - \left[\frac{d}{d-1}(\ell_t - 1)\cdot\exp(-k/d) + \frac{1}{d-1}(d-\ell_t)\cdot\exp(-k)\right] \\
&= (\ell_t - 1) - (\ell_t - 1)\frac{d}{d-1}\cdot\exp(-k/d) - \frac{1}{d-1}(d-(\ell_t-1)-1)\cdot\exp(-k) \\
&= (\ell_t - 1) - (\ell_t - 1)\frac{d}{d-1}\cdot\exp(-k/d) + (\ell_t-1)\frac{1}{d-1}\exp(-k) - (d-1)\frac{1}{d-1}\exp(-k) \\
&= (\ell_t - 1) - (\ell_t - 1)\frac{d}{d-1}\cdot\exp(-k/d) + (\ell_t-1)\frac{1}{d-1}\exp(-k) - \exp(-k) \\
&= (\ell_t - 1)\cdot\left[1 - \frac{d}{d-1}\cdot\exp(-k/d) + \frac{1}{d-1}\exp(-k)\right] - \exp(-k) \qquad (4.12)
\end{aligned}$$

As stated above, ℓ_t is implicitly bounded by the expansion. The classic vertex expansion α is defined to be

$$\alpha = \min_{0\leq|S|\leq n/2}\frac{|N(S)\setminus S|}{|S|}.$$

Recall that with our definitions, $N(S)$ is the *inclusive* neighbourhood of S, and the vertex expansion measures the ratio of the *exclusive* neighbourhood versus starting set. Therefore,

$$n_t/s_t = \ell_t \geq 1+\alpha \quad\Leftrightarrow\quad \ell_t - 1 \geq \alpha \qquad (4.13)$$

We now wish to replace the dependence in Eqn. (4.12) of μ on ℓ_t by one on α instead, using Eqn. (4.13). This gives us

$$\mu \leq \alpha\cdot\left[1 - \frac{d}{d-1}\cdot\exp(-k/d) + \frac{1}{d-1}\exp(-k)\right] - \exp(-k). \qquad (4.14)$$

We define $f(\alpha, d, k)$ to be the RHS of Eqn. (4.14). For there to be the possibility of finding a constant $\mu > 0$ we need to have $f(\alpha, d, k) > 0$ as well, that is

$$\alpha\cdot\left[1 - \frac{d}{d-1}\cdot\exp(-k/d) + \frac{1}{d-1}\exp(-k)\right] - \exp(-k) > 0$$

and, equivalently,

$$\begin{aligned}\alpha &> \frac{\exp(-k)}{1-\frac{d}{d-1}\cdot\exp(-k/d)+\frac{1}{d-1}\exp(-k)} \\ &= \frac{\exp(-k)}{\frac{d-1}{d-1}-\frac{d}{d-1}\cdot\exp(-k/d)+\frac{1}{d-1}\exp(-k)} \\ &= \frac{(d-1)\cdot\exp(-k)}{(d-1)-d\cdot\exp(-k/d)+\exp(-k)}\end{aligned} \tag{4.15}$$

Eqn. (4.15) now imposes a lower bound on the vertex expansion onto our set-up. Let's define $a(d,k)$ to be the RHS of Eqn. (4.15):

$$a(d,k) = \frac{(d-1)\cdot\exp(-k)}{(d-1)-d\cdot\exp(-k/d)+\exp(-k)}.$$

It is known [103] that for any $d \geq 3$, almost all d-regular graphs have edge expansion at least $c_d \cdot d$ (if nd is even), where c_d depends only on d and not on n. In fact, $c_d \geq 0.18$ for all $d \geq 3$, and $c_d \to 1/2$ as $d \to \infty$. Furthermore, whenever a graph has *edge* expansion ϕ, then from Eqn.(4.4) its *vertex* expansion is at least ϕ/d, implying that almost all d-regular graphs ($d \geq 3$) have vertex expansion at least 0.18.

It is easy to verify that $a(d,k) \leq 0.18$ for all $d \geq 3$ and $k = k(d) \geq d$. Note that we can also show that $a(d,k) \leq 0.18$ when $k \geq \log(d)$ and a large d.

It follows that $a(d,k) \leq 0.18 \leq \alpha$, and almost all d-regular graphs satisfy Eqn. (4.15).

The vertex expansion property holds true for any subset of size at most $n/2$. This implies that $\mathbb{E}[s_{t+1}] \geq (1+\mu)s_t$, where $\mu = \min_{\alpha,d,k} f(\alpha,d,k)/2$, so long as $s_t \leq n/2$. Using the same arguments as in Lemma 13. and Lemma 14. in [4], it follows that, with probability $1-n^{-c}$, there are βn active nodes after $\Theta(\log n)$ many steps.

□

In the next lemma we show that, given more than half of the nodes are active, the PULL COBRA protocol can inform all the nodes in $O(\log n)$ time.

Lemma 4.11. *Let $G(V,E)$ be a random d-regular graph. Assume there are constants $\beta > 1/2$ such that at least βn are active. Then $O(\log n)$ rounds of the* PULL COBRA *protocol inform all nodes of G,* w.h.p.

Proof. Let S be the set of active nodes, where $|S| \geq \beta \cdot n > n/2$. We want to prove that the COBRA PULL protocol will cover any vertex $v \in V \backslash S$ in $O(\log n)$ time. For every $v \in V \backslash S$, consider an instance of COBRA PUSH starting from v and running for t time

steps. This process can be described as a tree (a directed acyclic graph) rooted from v. An edge (u, w) is in the tree if and only if u sends a message to a node w via the COBRA PUSH process within time t. Note that external nodes of the tree are active nodes of the COBRA PUSH process after t steps. Now, let I_v be the set of external nodes of the tree after $t_1 = O(\log n)$ rounds of COBRA PUSH. According to Lemma 4.10, when $k \geq d$, $I_v \geq n/2 + 1$ with probability $1 - n^{-c}$. From the pigeon-hole principle, we know that there exists a vertex s_v such that $s_v \in S \cap I_v$. Now, let p_{t_1} be the probability of reaching s_v from v, via the COBRA PUSH strategy, after $t_1 = O(\log n)$. By Lemma 4.12, p_{t_1} is also the probability of the event in which we start from s_v and reach v, via the COBRA PULL strategy, after $t_1 = O(\log n)$. We know $p_{t_1} \geq 1 - n^{-c}$. Therefore, we inform all vertices of $V \backslash S$ from S, with probability at least $(1 - n^{-c})^{|V \backslash S|} \geq (1 - n^{-c})^{n/2}$.

□

Now, we show a lemma which compares the probability that a message is sent from node u to v via COBRA PUSH with the probability that it is sent from u to v via COBRA PULL. We use similar arguments in Lemma 4.1 of [68]

Lemma 4.12. *Let $\mathcal{E}_{C\text{PUSH}}(v, u, t)$ be the event that vertex u learns a rumour from vertex v after t rounds of the* PUSH COBRA *algorithm; and let $\mathcal{E}_{C\text{PULL}}(v, u, t)$ be the same event under the* PULL COBRA *algorithm. Then, $Pr(\mathcal{E}_{C\text{PUSH}}(v, u, t)) = Pr(\mathcal{E}_{C\text{PULL}}(u, v, t))$.*

Proof. Let $\mathcal{E}_{\text{PUSH}}(u, v, t)$ ($\mathcal{E}_{\text{PULL}}(u, v, t)$) be the event that COBRA PUSH (COBRA PULL) starting from a single node u informs a node v after exactly t rounds. The execution of COBRA PUSH (PULL) can be described as a directed acyclic graph (DAG). Nodes of the DAG are labelled with the IDs of nodes in G. Note that labels are not necessarily unique. Edges represent the random choices of the algorithm. For COBRA PUSH, an edge (u, v) is in the DAG if and only if node u sends the message to a node v.

A node v makes a successful request if it sends a request to an active node. For COBRA PULL, an edge (u, v) is in the DAG if and only if node v makes a successful request to a node u.

Recall by definition a node is active in timestep t if and only if it receives the message in timestep $t - 1$. The events $\mathcal{E}_{\text{PUSH}}(u, v, t)$ ($\mathcal{E}_{\text{PULL}}(u, v, t)$) can therefore be described by a path of length t. We will describe an edge as being activated if and only if one or more messages traverses it. Let $p_t := u, u_1, u_2, \ldots, u_{t-1}, v$ be a path of length t that satisfies $\mathcal{E}_{\text{PUSH}}(u, v, t)$. The probability that an edge (u_{i-1}, u_i) is not activated using COBRA PUSH, given that u_{i-1} is active, is $(1 - 1/d)^k$. Therefore the probability that the edge (u_{i-1}, u_i) is activated using COBRA PUSH given that u_{i-1} is active is $1 - (1 - 1/d)^k$.

Let the path $p'_t := v, v_1, v_2, \ldots, v_{t-1}, u$ be such that it satisfies $\mathcal{E}_{\text{PULL}}(v, u, t)$. Recall that (v, v_1) is an edge if v_1 makes a successful request to v. The probability that an edge (v_{i-1}, v_i) exists is $1-(1-1/d)^k$ for similar reasons to those used above for COBRA PUSH. The probability of the path described occurring is given by the product of the above probabilities.

Given that the paths described are unique we can conclude that

$$Pr\left(\mathcal{E}_{\text{PULL}}(v, u, t)\right) = Pr\left(\mathcal{E}_{\text{PUSH}}(u, v, t)\right).$$

□

Using previous lemmas, we are now ready to prove the main result of this section Theorem 4.9.

Proof of Theorem 4.9. Consider the case where we start from an arbitrary active vertex $u \in G$. Consider an instance of COBRA PUSH starting from vertex u. According to Lemma 4.10, this instance of COBRA PUSH (when $k \geq d$) will reach an active set S, where $S \geq n/2 + 1$, with probability at least $1 - n^{-c}$ after $t_1 = O(\log n)$ time. By Lemma 4.11, after (extra) $t_1 = O(\log n)$ rounds of the COBRA PULL strategy, we inform every $v \in V \backslash S$ with probability at least $(1 - n^{-c})^{|V \backslash S|}$. Hence, starting from u, after $O(\log n)$ rounds of COBRA PUSH & PULL strategy, we inform all the vertices with probability at least $(1 - n^{-c})^{|V \backslash S|+1} \geq (1 - n^{-c})^{n/2}$. This completes the proof. □

The proof of Theorem 4.8 trivially follows from the above lemmas and theorem when the branching factor k is sufficiently large ($k \geq d$).

4.6 Simulations

Although our analysis of the COBRA walk requires a branching factor k to be larger than d, in practice it does not need to be that large. Figure 4.1 shows the percentage of *active* and *covered* nodes of a 10-regular graph with 1 million nodes as a function of time. Here, for different values of $k = 2, 6 < 10$ the algorithm performs well.

4.7 Conclusion

In this chapter we analysed the so called COBRA walk introduced in [4]. We modelled the COBRA walk in a rumour spreading framework, where we examined it as a modification

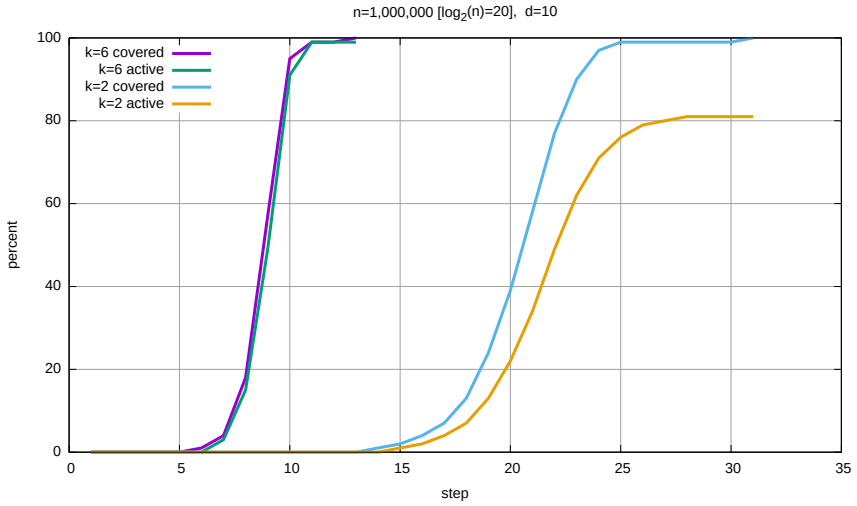

FIGURE 4.1: Percentage of active/covered nodes for $k = 2, 6$ as a function of time

of the PUSH strategy. Specifically, we improve the cover time of the COBRA walk for random regular graphs. Here, we show that it is possible to reduce the broadcasting time of the cobra walk from $\log^2(n)$ to $\log(n)$ at the cost of having to increase the branching factor k. However, the branching factor can be reduced to $\log(d)$ when d is sufficiently large.

Bibliography

[1] P. Berenbrink, T. Friedetzky, F. Mallmann-Trenn, S. Meshkinfamfard, and C. Wastell. Threshold load balancing with weighted tasks. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 550–558, May 2015. doi: 10.1109/IPDPS.2015.73.

[2] S. Hoefer and T. Sauerwald. Threshold load balancing in networks. *CoRR*, abs/1306.1402, 2013. URL http://arxiv.org/abs/1306.1402.

[3] Heiner Ackermann, Simon Fischer, Martin Hoefer, and Marcel Schöngens. Distributed algorithms for qos load balancing. *Distributed Computing*, 23(5-6):321–330, 2011. doi: 10.1007/s00446-010-0125-1. URL http://dx.doi.org/10.1007/s00446-010-0125-1.

[4] Chinmoy Dutta, Gopal Pandurangan, Rajmohan Rajaraman, and Scott Roche. Coalescing-branching random walks on graphs. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '13, pages 176–185, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1572-2. doi: 10.1145/2486159.2486197. URL http://doi.acm.org/10.1145/2486159.2486197.

[5] H. Hexmoor. *Computational Network Science: An Algorithmic Approach*. Computer Science Reviews and Trends. Elsevier Science, 2014. ISBN 9780128011560. URL https://books.google.co.uk/books?id=ID7LAwAAQBAJ.

[6] ITU. Itu releases 2014 ict figures. 2014. URL http://www.itu.int/net/pressoffice/press_releases/2014/23.aspx#.VjH94kISeUl.

[7] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pages 1–12, New York, NY, USA, 1987. ACM. ISBN 0-89791-239-X. doi: 10.1145/41840.41841. URL http://doi.acm.org/10.1145/41840.41841.

[8] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 2006. ISSN 1063-6692. doi: 10.1109/TIT.2006.874516. URL http://dx.doi.org/10.1109/TIT.2006.874516.

[9] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger. *Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-Tolerance (Texts in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. ISBN 3540008462.

[10] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1455–1466 vol. 2, March 2005. doi: 10.1109/INFCOM.2005.1498374.

[11] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumor spreading in social networks. *Theoretical Computer Science*, 412(24):2602 – 2610, 2011. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.tcs.2010.11.001. URL http://www.sciencedirect.com/science/article/pii/S0304397510006122. Selected Papers from 36th International Colloquium on Automata, Languages and Programming (ICALP 2009).

[12] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. ISBN 9780511814075. URL http://dx.doi.org/10.1017/CBO9780511814075. Cambridge Books Online.

[13] László Babai. Monte-carlo algorithms in graph isomorphism testing. *Université tde Montréal Technical Report, DMS*, pages 79–10, 1979.

[14] R. Solovay and V. Strassen. A fast monte-carlo test for primality. *SIAM Journal on Computing*, 6(1):84–85, 1977. doi: 10.1137/0206006. URL http://dx.doi.org/10.1137/0206006.

[15] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128 – 138, 1980. ISSN 0022-314X. doi: http://dx.doi.org/10.1016/0022-314X(80)90084-0. URL http://www.sciencedirect.com/science/article/pii/0022314X80900840.

[16] Richard M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, 34(1–3):165 – 201, 1991. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/0166-218X(91)90086-C. URL http://www.sciencedirect.com/science/article/pii/0166218X9190086C.

[17] P. A. P. Moran. Random processes in genetics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 54:60–+, 1958. URL http://adsabs.harvard.edu/cgi-bin/nph-data_query?bibcode=1958MPCPS..54...60M&link_type=EJOURNAL.

[18] Hauert Ch. E. and Nowak M. Lieberman. Evolutionary dynamics on graphs. *Nature*, 433:312–316, 2005.

[19] George B. Mertzios and Paul G. Spirakis. Strong bounds for evolution in networks. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *ICALP (2)*, volume 7966 of *Lecture Notes in Computer Science*, pages 669–680. Springer, 2013. ISBN 978-3-642-39211-5. URL http://dblp.uni-trier.de/db/conf/icalp/icalp2013-2.html#MertziosS13.

[20] Berthold Vöcking. Selfish load balancing. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, pages 517–542. Cambridge University Press, 2007. ISBN 9780511800481. URL http://dx.doi.org/10.1017/CBO9780511800481.022. Cambridge Books Online.

[21] Martin Raab and Angelika Steger. "balls into bins" - a simple and tight analysis. In *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, RANDOM '98, pages 159–170, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-65142-X. URL http://dl.acm.org/citation.cfm?id=646975.711521.

[22] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, September 1999. ISSN 0097-5397. doi: 10.1137/S0097539795288490. URL http://dx.doi.org/10.1137/S0097539795288490.

[23] Petra Berenbrink, Friedhelm Meyer auf der Heide, and Klaus Schröder. Allocating weighted jobs in parallel. In *Proceedings of the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '97, pages 302–310, New York, NY, USA, 1997. ACM. ISBN 0-89791-890-8. doi: 10.1145/258492.258522. URL http://doi.acm.org/10.1145/258492.258522.

[24] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: The heavily loaded case. *SIAM J. Comput.*, 35(6):1350–1385, 2006. doi: 10.1137/S009753970444435X. URL http://dx.doi.org/10.1137/S009753970444435X.

[25] Peter Sanders. On the competitive analysis of randomized static load balancing.

[26] Kunal Talwar and Udi Wieder. Balanced allocations: The weighted case. In *ACM Symposium on Theory of Computing (STOC)*. Association for Computing

Machinery, Inc., June 2007. URL http://research.microsoft.com/apps/pubs/default.aspx?id=66806.

[27] Paul W. Goldberg. Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, PODC '04, pages 131–140, New York, NY, USA, 2004. ACM. ISBN 1-58113-802-4. doi: 10.1145/1011767.1011787. URL http://doi.acm.org/10.1145/1011767.1011787.

[28] Eyal Even-Dar and Yishay Mansour. Fast convergence of selfish rerouting. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 772–781, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. ISBN 0-89871-585-7. URL http://dl.acm.org/citation.cfm?id=1070432.1070541.

[29] P. Berenbrink, T. Friedetzky, L. Goldberg, P. Goldberg, Z. Hu, and R. Martin. Distributed selfish load balancing. *SIAM J. Comput.*, 37(4):1163–1181, 2007. doi: 10.1137/060660345. URL http://dx.doi.org/10.1137/060660345.

[30] P. Berenbrink, M. Hoefer, and T. Sauerwald. Distributed selfish load balancing on networks. *ACM Transactions on Algorithms*, 11(1):2, 2014. doi: 10.1145/2629671. URL http://doi.acm.org/10.1145/2629671.

[31] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. *Random Struct. Algorithms*, 13(2):159–188, 1998. doi: 10.1002/(SICI)1098-2418(199809)13:2⟨159::AID-RSA3⟩3.0.CO;2-Q. URL http://dx.doi.org/10.1002/(SICI)1098-2418(199809)13:2<159::AID-RSA3>3.0.CO;2-Q.

[32] P. Berenbrink, K. Khodamoradi, T. Sauerwald, and A. Stauffer. Balls-into-bins with nearly optimal load distribution. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013*, pages 326–335, 2013. doi: 10.1145/2486159.2486191. URL http://doi.acm.org/10.1145/2486159.2486191.

[33] V. Stemann. Parallel balanced allocations. In *Proceedings of the Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '96, pages 261–269, New York, NY, USA, 1996. ACM. ISBN 0-89791-809-6. doi: 10.1145/237502.237565. URL http://doi.acm.org.proxy.lib.sfu.ca/10.1145/237502.237565.

[34] P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin. On weighted balls-into-bins games. *Theor. Comput. Sci.*, 409(3):511–520, 2008. doi: 10.1016/j.tcs.2008.09.023. URL http://dx.doi.org/10.1016/j.tcs.2008.09.023.

[35] K. Talwar and U. Wieder. Balanced allocations: the weighted case. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 256–265, 2007. doi: 10.1145/1250790.1250829. URL http://doi.acm.org/10.1145/1250790.1250829.

[36] Yuval Peres, Kunal Talwar, and Udi Wieder. The (1 + beta)-choice process and weighted balls-into-bins. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1613–1619, 2010. doi: 10.1137/1.9781611973075.131. URL http://dx.doi.org/10.1137/1.9781611973075.131.

[37] P. Berenbrink, T. Friedetzky, I. Hajirasouliha, and Z Hu. Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks. *Algorithmica*, 62 (3-4):767–786, 2012. doi: 10.1007/s00453-010-9482-1. URL http://dx.doi.org/10.1007/s00453-010-9482-1.

[38] László Lovász. Random walks on graphs: A survey. 1993.

[39] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. Providence, R.I. American Mathematical Society, 2009. ISBN 978-0-8218-4739-8. URL http://opac.inria.fr/record=b1128575. With a chapter on coupling from the past by James G. Propp and David B. Wilson.

[40] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995. ISBN 0-521-47465-5, 9780521474658.

[41] B. Doerr and S. Pohl. Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In *Genetic and Evolutionary Computation Conference, GECCO '12, Philadelphia, PA, USA, July 7-11, 2012*, pages 1317–1324, 2012. doi: 10.1145/2330163.2330346. URL http://doi.acm.org/10.1145/2330163.2330346.

[42] Easley David and Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010. ISBN 0521195330, 9780521195331.

[43] Josef Hofbauer and Karl Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[44] Michihiro Kandori, George J. Mailath, and Rafael Rob. Learning, mutation, and long run equilibria in games. *Ecta*, 61(1):29–56, Jan. 1993. ISSN 00129682.

[45] William H. Sandholm. *Population Games And Evolutionary Dynamics*. Economic learning and social evolution. MIT Press, 2010. ISBN 978-0-262-19587-4.

[46] Herbert Gintis. *Game theory evolving : a problem-centered introduction to modeling strategic interaction*. Princeton University Press, 2009. ISBN 9780691140506 0691140502 9780691140513 0691140510.

[47] H Ohtsuki and M Nowak. Evolutionary games on cycles. *P R Soc B*, 273(1598): 2249–2256, 2006.

[48] Arne Traulsen and Christoph Hauert. *Stochastic Evolutionary Game Dynamics*, pages 25–61. Wiley-VCH Verlag GmbH & Co. KGaA, 2010. ISBN 9783527628001. doi: 10.1002/9783527628001.ch2. URL http://dx.doi.org/10.1002/9783527628001.ch2.

[49] Christine Taylor, Yoh Iwasa, and Martin A. Nowak. A symmetry of fixation times in evoultionary dynamics. *Journal of Theoretical Biology*, 243(2):245 – 251, 2006. ISSN 0022-5193. doi: http://dx.doi.org/10.1016/j.jtbi.2006.06.016. URL http://www.sciencedirect.com/science/article/pii/S0022519306002529.

[50] Christine Taylor, Drew Fudenberg, Martin A. Nowak, and et al. Evolutionary game dynamics in finite populations, 2004.

[51] Richard M. Karp and Michael Luby. Monte-carlo algorithms for enumeration and reliability problems. In *FOCS*, pages 56–64. IEEE Computer Society, 1983. ISBN 0-8186-0508-1. URL http://dblp.uni-trier.de/db/conf/focs/focs83.html#KarpL83.

[52] Jan Rychtář and Brian Stadler. Evolutionary dynamics on small-world networks. *World Academy of Science and Engineering and Technology*, 2(7):1019 – 1022, 2008. ISSN 1307-6892.

[53] T. Antal and I. Scheuring. Fixation of strategies for an evolutionary game in finite populations. *B Math Biol*, 68(8):1923–44, 2006.

[54] Martin A. Nowak. *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press, 2006.

[55] George Giakkoupis and Thomas Sauerwald. Rumor spreading and vertex expansion. In Yuval Rabani, editor, *SODA*, pages 1623–1641. SIAM, 2012. ISBN 978-1-61197-309-9. URL http://dblp.uni-trier.de/db/conf/soda/soda2012.html#GiakkoupisS12.

[56] Petra Berenbrink, Robert Elsaesser, and Tom Friedetzky. Efficient randomised broadcasting in random regular networks with applications in peer-to-peer systems. In *Proceedings of the Twenty-seventh ACM Symposium on Principles of Distributed Computing*, PODC '08, pages 155–164, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-989-0. doi: 10.1145/1400751.1400773. URL http://doi.acm.org/10.1145/1400751.1400773.

[57] Damon Mosk-Aoyama and Devavrat Shah. Computing separable functions via gossip. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 113–122, New York, NY, USA, 2006. ACM. ISBN 1-59593-384-0. doi: 10.1145/1146381.1146401. URL http://doi.acm.org/10.1145/1146381.1146401.

[58] M Broom and J Rychtář. An analysis of the fixation probability of a mutant on special classes of non-directed graphs. *Proceedings A: Mathematical, Physical and Engineering Sciences*, 464(2098):2609–2627, October 2008. URL http://sro.sussex.ac.uk/30050/.

[59] Josep Díaz, Leslie Ann Goldberg, George B. Mertzios, David Richerby, Maria J. Serna, and Paul G. Spirakis. Approximating fixation probabilities in the generalized moran process. *Algorithmica*, 69(1):78–91, 2014. URL http://dblp.uni-trier.de/db/journals/algorithmica/algorithmica69.html#DiazGMRSS14.

[60] MR Frean and GJ Baxter. Death-birth ordering and suppression of fitness in networks. Technical report, Working paper: http://homepages. mcs. vuw. ac. nz/marcus/manuscripts/FreanBaxterJTB. pdf, 2008.

[61] Hisashi Ohtsuki and Martin A. Nowak. The replicator equation on graphs. *Journal of Theoretical Biology*, 243(1):86 – 97, 2006. ISSN 0022-5193. doi: http://dx.doi.org/10.1016/j.jtbi.2006.06.004. URL http://www.sciencedirect.com/science/article/pii/S0022519306002426.

[62] Kamran Kaveh, Natalia L. Komarova, and Mohammad Kohandel. The duality of spatial death–birth and birth–death processes and limitations of the isothermal theorem. *Royal Society Open Science*, 2(4), 2015. doi: 10.1098/rsos.140465.

[63] Laura Hindersin and Arne Traulsen. Most undirected random graphs are amplifiers of selection for birth-death dynamics, but suppressors of selection for death-birth dynamics. *PLoS Comput Biol*, 11(11):e1004437, 11 2015. doi: 10.1371/journal.pcbi.1004437. URL http://dx.doi.org/10.1371%2Fjournal.pcbi.1004437.

[64] Christine Taylor, Drew Fudenberg, Akira Sasaki, and MartinA. Nowak. Evolutionary game dynamics in finite populations. *Bulletin of Mathematical Biology*, 66(6):1621–1644. ISSN 0092-8240. URL http://dx.doi.org/10.1016/j.bulm.2004.03.004.

[65] S. Karlin and H.E. Taylor. *A First Course in Stochastic Processes*. Elsevier Science, 2012. ISBN 9780080570419. URL https://books.google.co.uk/books?id=dSDxjX9nmmMC.

[66] Colin Cooper, Alan Frieze, and Tomasz Radzik. Multiple random walks in random regular graphs. *SIAM J. Discret. Math.*, 23(4):1738–1761, November 2009. ISSN 0895-4801. doi: 10.1137/080729542. URL http://dx.doi.org/10.1137/080729542.

[67] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many random walks are faster than one. In *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '08, pages 119–128, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-973-9. doi: 10.1145/1378533.1378557. URL http://doi.acm.org/10.1145/1378533.1378557.

[68] George Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 57–68, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-25-5. doi: http://dx.doi.org/10.4230/LIPIcs.STACS.2011.57. URL http://drops.dagstuhl.de/opus/volltexte/2011/2997.

[69] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Almost tight bounds for rumour spreading with conductance. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 399–408, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0050-6. doi: 10.1145/1806689.1806745. URL http://doi.acm.org/10.1145/1806689.1806745.

[70] D.J. Aldous. On the time taken by random walks on finite groups to visit every state. *Z. Wahrsch. Verw. Gebiete*, 62:361–374, 1983.

[71] Colin Cooper and Alan Frieze. The cover time of sparse random graphs. *Random Struct. Algorithms*, 30(1-2):1–16, 2007. URL http://dblp.uni-trier.de/db/journals/rsa/rsa30.html#CooperF07.

[72] Andrei Z. Broder and Anna R. Karlin. Bounds on the cover time (preliminary version). In *FOCS*, pages 479–487. IEEE Computer Society, 1988. ISBN 0-8186-0877-3. URL http://dblp.uni-trier.de/db/conf/focs/focs88.html#BroderK88.

[73] Ashok K. Chandra, Prabhakar Raghavan, Walter L. Ruzzo, Roman Smolensky, and Prasoon Tiwari. The electrical resistance of a graph captures its commute and cover times (detailed abstract). In David S. Johnson, editor, *STOC*, pages 574–586. ACM, 1989. ISBN 0-89791-307-8. URL http://dblp.uni-trier.de/db/conf/stoc/stoc89.html#ChandraRRST89.

[74] Johan Jonasson. On the cover time for random walks on random graphs. *Combinatorics, Probability and Computing*, 7(3):265–279, 1998. URL http://dblp.uni-trier.de/db/journals/cpc/cpc7.html#Jonasson98.

[75] J. Jonasson and O. Schramm. On the cover time of planar graphs. *ArXiv Mathematics e-prints*, February 2000.

[76] Uriel Feige. A tight lower bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(4):433–438, 1995. URL http://dblp.uni-trier.de/db/journals/rsa/rsa6.html#Feige95a.

[77] Greg Barnes and Uriel Feige. Short random walks on graphs. *SIAM J. Discrete Math.*, 9(1):19–28, 1996. URL http://dblp.uni-trier.de/db/journals/siamdm/siamdm9.html#BarnesF96.

[78] Andrei Z. Broder. Generating random spanning trees. In *FOCS*, pages 442–447. IEEE Computer Society, 1989. ISBN 0-8186-1982-1. URL http://dblp.uni-trier.de/db/conf/focs/focs89.html#Broder89.

[79] Robert Elsässer and Thomas Sauerwald. Tight bounds for the cover time of multiple random walks. *Theor. Comput. Sci.*, 412(24):2623–2641, 2011. URL http://dblp.uni-trier.de/db/journals/tcs/tcs412.html#ElsasserS11.

[80] Alan J. Demers, Daniel H. Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard E. Sturgis, Daniel C. Swinehart, and Douglas B. Terry. Epidemic algorithms for replicated database maintenance. In Fred B. Schneider, editor, *PODC*, pages 1–12. ACM, 1987. ISBN 0-89791-239-X. URL http://dblp.uni-trier.de/db/conf/podc/podc87.html#DemersGHIL87.

[81] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Randomized broadcast in networks. *Random Struct. Algorithms*, 1(4):447–460, 1990. URL http://dblp.uni-trier.de/db/journals/rsa/rsa1.html#FeigePRU90.

[82] R. Elsässer, U. Lorenz, and T. Sauerwald. On randomized broadcasting in star graphs. *Discrete Applied Mathematics*, 157(1):126 – 139, 2009. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2008.05.018. URL http://www.sciencedirect.com/science/article/pii/S0166218X08002084.

[83] Robert Elsässer and Thomas Sauerwald. Broadcasting vs. mixing and information dissemination on cayley graphs. In *In Proc. of STACS'07*, 2007.

[84] Nikolaos Fountoulakis, Anna Huber, and Konstantinos Panagiotou. Reliable broadcasting in random networks and the effect of density. In *Proceedings of the 29th Conference on Information Communications*, INFOCOM'10, pages 2552–2560, Piscataway, NJ, USA, 2010. IEEE Press. ISBN 978-1-4244-5836-3. URL http://dl.acm.org/citation.cfm?id=1833515.1833846.

[85] N. Fountoulakis and K. Panagiotou. Rumor Spreading on Random Regular Graphs and Expanders. *ArXiv e-prints*, February 2010.

[86] Robert Elsässer and Thomas Sauerwald. The power of memory in randomized broadcasting. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 218–227, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. URL http://dl.acm.org/citation.cfm?id=1347082.1347107.

[87] Robert Elsässer. On the communication complexity of randomized broadcasting in random-like graphs. In *SPAA 2006: Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Cambridge, Massachusetts, USA, July 30 - August 2, 2006*, pages 148–157, 2006. doi: 10.1145/1148109.1148135. URL http://doi.acm.org/10.1145/1148109.1148135.

[88] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized rumor spreading. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 565–574, 2000. doi: 10.1109/SFCS.2000.892324.

[89] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumour spreading and graph conductance. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 1657–1663, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. ISBN 978-0-898716-98-6. URL http://dl.acm.org/citation.cfm?id=1873601.1873736.

[90] David A. Kessler. Epidemic size in the sis model of endemic infections. *Journal of Applied Probability*, 45(3):pp. 757–778, 2008. ISSN 00219002. URL http://www.jstor.org/stable/27595985.

[91] R. Michael Tanner. Explicit concentrators from generalized n-gons. *SIAM Journal on Algebraic Discrete Methods*, 5(3):287–293, 1984. doi: 10.1137/0605030. URL http://dx.doi.org/10.1137/0605030.

[92] Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *FOCS*, pages 218–223. IEEE Computer Society, 1979. URL http://dblp.uni-trier.de/db/conf/focs/focs79.html#AleliunasKLLR79.

[93] Uriel Feige. A tight lower bound on the cover time for random walks on graphs. In *Random Structures and Algorithms*, pages 433–438, 1994.

[94] Uriel Feige. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(1):51–54, January 1995. ISSN 1042-9832. doi: 10.1002/rsa.3240060106. URL http://dx.doi.org/10.1002/rsa.3240060106.

[95] Peter Matthews. Covering problems for brownian motion on spheres. *The Annals of Probability*, 16(1):pp. 189–199, 1988. ISSN 00911798. URL http://www.jstor.org/stable/2243894.

[96] Peter Matthews. Some sample path properties of a random walk on the cube. *Journal of Theoretical Probability*, 2(1):129–146, 1989. ISSN 0894-9840. doi: 10.1007/BF01048275. URL http://dx.doi.org/10.1007/BF01048275.

[97] Thomas Sauerwald and Alexandre Stauffer. *Rumor Spreading and Vertex Expansion on Regular Graphs*, chapter 37, pages 462–475. doi: 10.1137/1.9781611973082.37. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611973082.37.

[98] Béla Bollobás. *Random Graphs*. Cambridge University Press, second edition, 2001. ISBN 9780511814068. URL http://dx.doi.org/10.1017/CBO9780511814068. Cambridge Books Online.

[99] Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1(4):311 – 316, 1980. ISSN 0195-6698. doi: http://dx.doi.org/10.1016/S0195-6698(80)80030-8. URL http://www.sciencedirect.com/science/article/pii/S0195669880800308.

[100] Edward A Bender and E.Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296 – 307, 1978. ISSN 0097-3165. doi: http://dx.doi.org/10.1016/0097-3165(78)90059-6. URL http://www.sciencedirect.com/science/article/pii/0097316578900596.

[101] A. Steger and N. C. Wormald. Generating random regular graphs quickly. *Comb. Probab. Comput.*, 8(4):377–396, July 1999. ISSN 0963-5483. doi: 10.1017/S0963548399003867. URL http://dx.doi.org/10.1017/S0963548399003867.

[102] Doron Puder. Expansion of random graphs: new proofs, new results. *Inventiones mathematicae*, 201(3):845–908, 2015. ISSN 0020-9910. doi: 10.1007/s00222-014-0560-x. URL http://dx.doi.org/10.1007/s00222-014-0560-x.

[103] David Ellis. The expansion of random regular graphs. Lecture notes. https://www.dpmms.cam.ac.uk/~dce27/randomreggraphs3.pdf.