DISTRIBUTED ALGORITHMS FOR RESOURCE ALLOCATION AND ROUTING

by

Zengjian Hu

B.Sc., Beijing Institute of Technology, 1997

M.Sc., Institute of Computing Technology, Chinese Academy of Sciences, 2000

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
2007

APPROVAL

Name: Zengjian Hu

Degree: Doctor of Philosophy

Title of thesis: Distributed Algorithms for Resource Allocation and Routing

Examining Committee: Dr. Qianping Gu
Chair

Dr. Petra Berenbrink
Assistant Professor of Computing Science
Senior Supervisor

Dr. Binay Bhattacharya
Professor of Computing Science
Supervisor

Dr. Funda Ergün
Associate Professor of Computing Science
SFU Examiner

Dr. Robert Elsässer, University of Paderborn
Junior Professor of Computer Science
External Examiner

Date Approved: Apr. 26 / 2007

Abstract

In this thesis, we study distributed algorithms in the context of two fundamental problems in distributed systems, resource allocation and routing. Resource allocation studies how to distribute workload evenly to resources. We consider two different resource allocation models, the diffusive load balancing and the weighted balls-into-bins games. Routing studies how to deliver messages from source to destination efficiently. We design routing algorithms for broadcasting and gossiping in ad hoc networks.

Diffusive load balancing studies how nodes with initial tasks in a network balance their loads concurrently with all their neighbours. We propose a novel analytical method to deal with the concurrent load balancing actions, which are the major obstacle for the analysis. The idea is to first sequentialize the concurrent load balancing actions, analyze this sequential system instead, and then bound the gap between both. We analyze various diffusive load balancing algorithms using this idea.

The weighted balls-into-bins game studies how to evenly allocate a set of independent weighted balls into a set of bins. In particular, we consider two different scenarios, the static sequential game and the selfish reallocation game. In the static sequential game, balls come one after another and need to be allocated in such order. We study how the outcome of the game, the expected maximum load of any bin, is influenced by the game parameters such as the distribution of ball weights and the order that balls are allocated. In the selfish reallocation game, every ball has its own initial location. An iterative, selfish distributed reallocation algorithm is applied to balance the workload. We show bounds for the convergence time of the algorithm, which is the number of steps to reach (or get close to) some equilibrium state.

We study routing algorithms for broadcasting and gossiping in ad hoc networks. We consider the so-called "energy efficient" ad hoc network model. Our goal is to minimize

not only broadcasting/gossiping time, but also energy consumption, which is measured by the total number of sent messages. We present and analyze several energy efficient broadcasting/gossiping algorithms for both random and general ad hoc networks.

To Rong, Ben, Dad and Mum

"The important thing is not to stop questioning. Curiosity has its own reason for existing."

— *Albert Einstein*

Acknowledgments

It is finally the time to thank the many people who made this thesis possible. I wish I had said thanks to you in person; Forgive me if I did not. First of all, I wish to express my deep appreciation to my senior supervisor, Dr. Petra Berenbrink, for whom I had the privilege for being her first Ph.D graduate. Petra had been a wonderful advisor as well as a good friend. Throughout my stay in SFU, she had spent a great deal of efforts and money discussing research with me, correcting my writeups, sending me to conferences and so on. She had helped me to mature more than ever both scientifically and personally. Thank you, Petra!

I am also very grateful to my supervisor, Dr. Binay Bhattacharya. Every time when I encountered problems, Binay was always the first person I seek for advice. Binay, your kind help and warm encouragement had been important to me. Thanks!

I also wish to thank all my teachers in SFU. I am especially grateful to Dr. Funda Ergün, Dr. Martin Ester, Dr. Qian-Ping Gu, Dr. Art Liestman and Dr. Pavol Hell. Their broad knowledge, as well as kind personalities have greatly inspired me. I am also indebted to my co-authors, Dr. Colin Cooper, Dr. Tom Friedetzky, Dr. Leslie Ann Goldberg, Dr. Paul Goldberg and Dr. Russell Martin, for the good collaborations which lead to this thesis. Special thanks also goes to Dr. Robert Elsässer, who came all the way from Europe to attend my defence.

I would like to thank my fellow gradudate students in the Computing Science department for creating a fun environment. Life would not be as joyful as it was here without my great friends: Zhengbing Bian, Tugkan Batu, Yuanzhu Peter Chen, Qidan Cheng, Andrew Clement, Bradley Coleman, Lei Duan, Zhe Fang, Richard Frank, Byron Gao, Baohua Gu, Iman Hajirasouliha, Yuzhuang Hu, Mayu Ishida, Varun Jain, Hao Jiang, Wen Jin, Hossein Jowhari, Emre Karakoc, Chiyoko Kawano, Mike Letourneau, Yudong Liu, Cheng Lu, Flavia

Moser, Lawrence Ryan, Zhongmin Shi, Qiaosheng Shi, Evgeny Skvortsov, Ming Su, Yi Sun, Stephen Tse, Dan Wang, Feng Wang, Yang Wang, Yong Wang, Ning Wei, Weihua Xiong, Yinan Zhang, Zhong Zhang and Senqiang Zhou.

I thank my entire extended family for always being so supportive. My grandma, parents-in-laws, sister, my brother-in-laws, sister-in-law, have always been there to support me.

Finally, and most importantly, I wish to thank my family. Rong, my wife, is always there to support me. Her love has accompanied me to get through the hard times. This thesis would not have been here without her. This thesis is dedicated to her and our newborn son, Ben, who has given us great joy. Dad and mum, I can not thank you enough for bringing me to this world, raising me and educating me. I know you were always proud for every single achievement I made. This thesis is also dedidated to you.

Contents

List of Tables

List of Figures

List of Programs

Chapter 1

Introduction

A distributed system typically consists of a set of devices with limited computational power and restricted ability to communicate with its peers. The latter limits may include the number of "reachable" peers, the interconnection structure itself, the communication bandwidth, where these and possibly other relevant parameters may even vary over time. The reader may think of these devices as processors, workstations, PCs, laptops, printers, wireless-enabled hand-held computers or even cellular phones, sensors, and so on. One crucial, defining feature of this kind of system that we are interested in, is the absence of any central instance that would take care of organizing any aspect of the system.

In this thesis, we focus on one of the fundamental issues in the study of distributed systems, the design and analysis of distributed algorithms. The notion of a distributed algorithm, just as that for distributed systems, is overloaded with many meanings. We understand it to be an algorithm that runs on a parallel and distributed system as described above. We will refer to the devices as nodes, hinting at the fact that the communication structures of these systems are often modelled as graphs. We assume that when a distributed algorithm is performed, every participating node is running a copy of it concurrently and independently. Since the nodes (or the algorithms running on them) typically have to operate on limited information – for instance, the number of participants, or possibly information beyond what is known to direct neighbours may not be available – they will need to communicate in some way in order to coordinate their actions.

Distributed algorithms have been widely used in distributed systems in a variety of areas such as scientific computing, distributed information processing, telecommunication, and many more. For instance, modern parallel computers, sometimes consisting of thousands

of processors, run distributed algorithms to tackle scientific computing problems such as genome analysis. Today's airline companies, insurance agencies and banks build parallel file systems to ensure fast access of their customer information. Communication networks like the Internet or mobile ad hoc networks, need efficient communication protocols to enable users in different geographical areas to conveniently exchange information. All these applications critically rely on distributed algorithms. In this thesis, we consider two problems fundamental to distributed systems, *resource allocation* and *routing*. We focus on how to design and analyze distributed algorithms for these two problems.

Resource Allocation. Resource allocation studies how to distribute workload to resources. It is one of the central problems in many distributed systems. For instance, resource allocation is critical for massive parallel computers to achieve a high system throughput [46, 63]. Resource allocation is also essential to minimize the response time of distributed file systems [105, 104]. Throughout the thesis we quantify work in terms of (not necessarily equal-sized) tasks.

In the study of the resource allocation problem, we often use load vectors to represent the workload situation. The goal is to minimize some cost function that typically capture a notion of "balancedness" of the load vector. For different practical scenarios we can have different cost functions. For example, we can use cost functions that characterize the deviation between the maximum and the average loads [68, 13, 20], or the variance of the load vector [46, 31, 63], or the number of empty resources [21]. We can also distinguish between resource allocation models based on which party makes the resource allocation decisions. For instance, we can let the resources decide where the tasks should be located [46, 31, 63], or let tasks decide which resources to choose [68, 13, 20]. In particular, the tasks may be "selfish", in that they only try to optimize their own workload situations instead of the global workload situation [22, 59, 60].

We study two resource allocation models, the *neighbourhood load balancing* and the *balls-into-bins game*. In the neighbourhood load balancing model [46, 31, 63], every node is initially assigned some arbitrary workload. In order to achieve a balanced workload (or to minimize the variance of the load vector), only the nodes connected by a link may exchange tasks. There are mainly two neighbourhood load balancing approaches, *diffusion* and *dimension exchange*. In the diffusion approach, nodes can concurrently exchange tasks with all their neighbours, while in the dimension exchange approach, nodes can only exchange

tasks with one neighbour at a time. In the *balls-into-bins game* [68, 13, 20], we are given a set of independent tasks. Every task runs a distributed algorithm to allocate itself to some resource so that (typically) the maximum load in any resource is minimized. In this model, every task is allowed to query a small set of nodes to get some partial load information, but it is not allowed to communicate with other tasks or nodes.

Routing. Routing studies how to deliver messages from source to destination in a timely manner. It is also a central problem in distributed systems. Routing consists of two major tasks, path selection and scheduling. Path selection refers to selecting a path for each packet from its origin to its destination, while scheduling refers to arranging the movements of the packets along their paths to avoid contention.

There have been a wide variety of routing algorithms for different networks. For routing in packet-switched networks like the Internet, data is usually split up into packets each of which is labeled with the complete destination address and is routed individually. The objective is to deliver packets to the destination using as few steps as possible. For example, Leighton, Maggs and Richa [79] propose an algorithm that can route any set of packets c on any network in $O(c + d)$ steps using constant-size queues, where c is the congestion of the paths, d is the length of the longest path. For routing in wireless ad hoc or sensor networks, since both the wireless channel and the power supply are scarce resources, algorithm designers must take these two issues into account. For instance, Chen, Low, Chiang and Doyle [34] propose a joint design of congestion control, path selection and scheduling for wireless ad hoc networks in order to use wireless channels more efficiently. Cartigny, Simplot and Stojmenović [33] study empirically how to achieve the minimum energy consumption for broadcasting in ad hoc networks. In this thesis, we focus on routing in ad hoc networks.

This thesis studies distributed algorithms for both resource allocation and routing. For the resource allocation problem, we first analyze the diffusion model. We then study the weighted balls-into-bins game, where every ball is associated with some weight. We also design efficient distributed routing algorithms for ad hoc networks. In the rest of this introductory chapter, we further describe the problems we consider in this thesis. Section 1.1 and 1.2 introduce distributed load balancing and routing respectively. We summarize our contributions and give the thesis outline in Section 1.3.

1.1 Distributed Load Balancing

As we discussed, resource allocation is a fundamental problem for many distributed systems. In the following we further introduce two load balancing models, the *diffusive load balancing* and the *balls-into-bins games*.

Diffusive Load Balancing

In the diffusive load balancing model, every node is initially associated with a certain number of tasks and the overall workload can be arbitrarily unbalanced. In order to achieve load balancing, the nodes with higher load can send some tasks over to those nodes with lower load. Moreover, the number of tasks in the system is time-invariant, i.e., neither do new tasks appear, nor do existing ones disappear. In the diffusive load balancing algorithms (e.g., [46, 31, 63, 23]) it is assumed that all the nodes are connected by a network, and in each step only the neighbouring nodes can exchange tasks. Our goal is to distribute tasks as evenly as possible among nodes. We are particularly interested in the time it takes to reach (or come close to) the perfectly balanced state. Note that for diffusion protocols, nodes are allowed to transfer workload concurrently, which makes the analysis quite difficult. In Chapter 2, we propose a novel analytical technique to cope with this concurrency issue.

Weighted Balls-into-bins Games

In the balls-into-bins game [13, 20, 90]), every task needs to allocate itself to some resource efficiently. To study this problem we often denote tasks as balls and resources as bins. In particular, we study the weighted balls-into-bins games, where the balls (tasks) are associated with positive weights. In practice, the weight of a task represents the resource requirement of the task, i.e., memory or running time.

We study two different variations of balls-into-bins games. We first consider the *static sequential* game, where we allocate a set of balls coming one after another sequentially. A well-known method is to let every ball choose $d \geq 1$ bins uniformly at random, and allocate itself into a bin with the minimum load. If $d = 1$, the game is called *single-choice*, otherwise *multiple-choice*. We ask the following two natural questions. How does the weight distribution affect the outcome of the game, and how does the order in which we allocate balls affect the outcome of the game? We propose an indirect approach, which compares an arbitrary weighted system with its uniform counterpart that only consists of unit-size

balls and has the same total weight as the weighted system. For the comparison, we use a majorization technique similar to [13].

We then study the so-called *selfish reallocation* game, a model that recently received much attention in the theory community. This problem models the selfish users' behaviors in applications in which users share common resources such as server bandwidth. In such applications, every user is *selfish* in that it only tries to optimize its own situation, i.e., the cost incurred by its host resource, without trying to optimize the global situation. To analyze this problem we again model the resources as bins and the users as balls. We assume that initially every ball is allocated to some arbitrary bin. Afterwards, every ball migrates to a different bin according to the following natural distributed reallocation protocol. In each step, every ball picks one bin uniformly at random. It then compares the load of its current host bin with the load of the randomly chosen bin. If the load difference is above a certain threshold then the ball will migrate to the destination bin with a certain probability. In this protocol, balls behave selfishly in that they only try to minimize the loads of their own bins. Note also that ball migrations happen in parallel. Using game theoretic notion, when all the balls stop moving, the system is said to be in some *Nash equilibrium*. We show upper and lower bounds for the convergence time, i.e., the number of steps for the system to reach (or get close to) one of the Nash equilibria.

1.2 Routing in Ad Hoc Networks

The second part of the thesis studies routing in ad hoc networks. Even though the routing problem appears very different from the resource allocation problem studied in the first part of the thesis, they both are fundamental mechanisms for distributed systems and their analytical techniques share many similarities.

An ad hoc network is a communication network composed by a set of independent mobile devices connected through a wireless medium. The main advantage of ad hoc networks is that they can be easily deployed since they do not need any (wired) infrastructure. This advantage makes ad hoc networks very useful in military environments, for example, building survivable radio communication networks in battlefields. Ad hoc networks are also widely used in civil applications such as disaster recovery, home networks and personal area networks [99].

We study how to design efficient routing algorithms for *broadcasting* and *gossiping* in

ad hoc networks. For the broadcasting problem, one node of the network needs to send a message to all other nodes in the network. For the gossiping problem, every node of the network needs to send a message to every other node. Due to the lack of fixed infrastructure, the broadcasting/gossiping algorithms running on ad hoc networks have to be carried out in a decentralized manner.

In a well-accepted theoretical model of ad hoc networks, (e.g., [41, 51, 55, 72, 112, 7, 73]), it is assumed that every device has a fixed communication range and it can reach all the devices within that range. It is also assumed that there is only one communication channel, so that if there is more than one device transmitting at the same time, their messages "collide" and need to be resent. In this model, the goal is to minimize the *broadcasting/gossiping time*, that is, the number of time steps to achieve broadcasting/gossiping. In practice, since the mobile devices tend to be small and have only limited power supply, energy efficiency is another important issue for communication in ad hoc networks (e.g., [66, 85]). Thus, in Chapter 4, we propose a novel energy efficient model for ad hoc networks. Under this model, we propose and analyze several efficient broadcasting and gossiping algorithms that achieve the the minimum energy consumption, where the *energy consumption* is measured in terms of the number of messages (or transmissions) sent by each node.

1.3 Contributions and Organization

To summarize, this thesis makes the following contributions.

- In Chapter 2, we propose a novel analytical method for the diffusion model. The key idea is to sequentialize the concurrent load balancing actions and analyze this new sequentialized system, and then to bound the gap between both systems. We demonstrate the usefulness of this approach by analyzing various natural diffusion-type algorithms. Our results are similar to, or better than previously existing ones, while our proofs are significantly easier.

- Chapter 3 considers two different scenarios of the weighted balls-into-bins games. For the static sequential game, we study how the weight distribution, or the order to allocate balls, influence the outcomes of the game, in particular, the expected maximum load. First, using a majorization approach, we show that for the single-choice game, a more balanced weight distribution always yields a smaller expected maximum load.

We then show that the same argument does not hold in the multiple-choice game when we have large number of balls. Finally, we study how the order to allocate balls affects the expected maximum load.

For the selfish reallocation game, we prove upper and lower bounds for the convergence time, i.e., the number of steps for the system to converge (or get close) to some Nash equilibrium. For a system with m balls and n bins, we show an upper bound of $O(mn\Delta^3\epsilon^{-2})$ for the convergence time, where Δ is the maximum weight of tasks. Our analysis is based on the potential function technique. In addition, we prove a lower bound of $\Omega(m\Delta/\epsilon)$ for the convergence time. Next, we apply our technique to the uniform case and show that our algorithm converges to the (real) Nash equilibrium in $O(\log m + n \log n)$ steps w.h.p. We then combine our protocol and the protocol in [22] to obtain a $O(\log\log m + n \log n)$ convergence time w.h.p. Finally, we show the tightness of this result by proving a matching lower bound.

- In Chapter 4, we study radio broadcasting and gossiping algorithms in ad hoc networks. We propose a new *energy efficient* communication model, in which the *energy consumption* of an algorithm is measured in terms of the total number of messages (or transmissions) sent. We consider both random and general networks. For random networks, we propose a $O(\log n)$ broadcasting algorithm where every node transmits at most once and a $O(d \log n)$ gossiping algorithm using $O(\log n)$ messages per node. For general networks with known diameter D, we present a randomized broadcasting algorithm with optimal broadcasting time $O(D\log(n/D) + \log^2 n)$ that uses $O(\log^2 n/\log(n/D))$ transmissions per node in expectation. Our lower bound $\Omega(\log^2 n/\log(n/D))$ on the number of transmissions matches our upper bound for time-invariant distributions. We also demonstrate a tradeoff between these two objectives.

Chapter 2

Diffusive Load Balancing

In this chapter we focus on an important resource allocation model named the diffusive load balancing. We are given a network where nodes represent processing units, edges represent communication links. Initially, each node is associated with an arbitrary number of tasks. Then in each time step, the neighbouring nodes are allowed to concurrently exchange certain amount of tasks to achieve load balancing.

We propose a new proof technique to analyze the above procedure. The technique is designed to handle concurrent load balancing actions, which are often the main obstacle for the analysis. We demonstrate the usefulness of this technique by analyzing various natural diffusion algorithms. Our results are similar to, or better than, previously existing ones, while our proofs are significantly simpler. The key idea of our proof technique is to first sequentialize the original concurrent load transfers, analyze this new sequential system, and then to bound the gap between both the concurrent and sequential systems.

2.1 Introduction

A well-known model of resource allocation model is the *neighbourhood load balancing* problem. We have a set of identical nodes (processors) that are connected by a network. Initially, every node is associated with some number of tasks. The number of tasks in the system is time-invariant, i.e., neither do new tasks appear, nor do existing tasks disappear. In each step, only the neighbouring nodes (nodes connected by an edge) are allowed to exchange certain number of tasks to balance their loads. The goal is to distribute tasks as evenly as possible among nodes using minimum number of steps. In the following, the *load* of a node

at time t is defined as the number of tasks on that node at that time.

Neighbourhood load balancing approaches can be classified in many ways. We can distinguish between the *diffusion* and *dimension exchange* approaches. For diffusion approaches, each node is allowed to balance their loads *concurrently* with all its neighbours. For dimension exchange approaches, each node can only communicate with one of its neighbours. The neighbour can be chosen either in a round-robin fashion[46], or by randomly generating a matching of the underlying network at every time step [63]. We can also distinguish between *discrete* and *continuous* approaches. For discrete approaches, tasks can not split and nodes are only allowed to exchange integer number of tasks. For continuous approaches, tasks can be split into arbitrarily small pieces and nodes are allowed to exchange fractional amount of tasks.

So far, the analysis techniques for *diffusion* and *dimension exchange* approaches are quite different. The common technique to analyze dimension exchange approaches is the potential function technique [46, 63, 62, 94]. In this technique, first, we choose a suitable potential function to measure the distance between a particular system state and the perfectly balanced state. We then show that the potential always decreases by a certain amount in every time step. However, it is challenging to use this technique for diffusion-type algorithms (see [62, 94]). The major challenge is that in the diffusion approaches, loads can be transferred concurrently so that the load situation could change drastically in one step. This makes it difficult to apply the potential function technique to analyze diffusion approaches. So far the common technique to analyze diffusion approaches is the algebraic technique (see [46, 107, 94]), which only works for the continuous case. See Section 2.2 for an overview.

In this work we propose a simple potential function technique to analyze diffusion load balancing approaches. The idea is as follows. First, we "sequentialize" the concurrent load balancing actions of the diffusion approach to get a sequential system. The technique to sequentialize the concurrent load balancing actions will be explained later in Section 2.4. Since there is no concurrent load balancing action, the sequential system can be easily analyzed using existing ideas, e.g., the one from [63]. We then use the potential drop in the sequential system to lower bound the potential drop in the original concurrent system. In more detail, we show that under certain conditions, the potential drops in both the sequentialized and concurrent systems at most differ by a constant factor.

We use this technique to analyze the standard diffusion algorithm in the continuous and discrete cases. Then we apply our technique to get results for the dynamic model of [58],

where the network can change over time, for both cases. Finally, other than the traditional neighbourhood load balancing approaches, we also consider a randomized approach which allows nodes to randomly choose their load balancing partners among the set of all other nodes. Note that in this setting, a node can easily be forced to balance its load with many other nodes, so that many concurrent load balancing actions will take place. Note also that this setting can be regarded as neighbourhood load balancing where the network topology is randomly chosen and changes from step to step. We call such a network *random* in the following.

We organize the rest of this chapter as follows. We review related work in Section 2.2. Section 2.3 consists of our model and new results. Section 2.4, 2.5 and 2.6 study the diffusion approaches on fixed network, dynamic network and random network. Finally we conclude in Section 2.7.

2.2 Related Work

In this section we review related work. We discuss both *continuous* and *discrete* load balancing approaches.

2.2.1 Continuous Load Balancing

Continuous load balancing is the "ideal" case in which tasks can be split arbitrarily. Hence it is possible to balance the workload perfectly. In the following we review diffusion and dimension exchange approaches.

Diffusion. Cybenko [46] and, independently, Boillat [31], are the first to study the diffusion approaches. In the diffusion model of Cybenko, the load distribution at time step t is quantified by an n vector, $L^t = (\ell_1^t, \ldots, \ell_n^t)$, where ℓ_i^t is the load of node i at time $t \geq 0$. In each round t, node i and node j compare their load. If $\ell_j^t > \ell_i^t$, node j sends $\alpha\left(\ell_j^t - \ell_i^t\right)$ tasks to node i. α is called the *diffusion factor* and is set to $1/(\delta + 1)$, where δ is the maximum degree of the network. We can write $L^{t+1} = M \cdot L^t$, where $M = (m_{ij})$ is a matrix defined as

$$m_{ij} = \begin{cases} \alpha_{ij}, & \text{if } i \neq j \\ 1 - \sum_k \alpha_{ik} & \text{if } i = j. \end{cases}$$

M is commonly referred to as *diffusion matrix*. Cybenko [46] (see also [107, 94]) shows a tight connection between the convergence rate of his diffusion algorithm and the second largest eigenvalue of M. Let $\bar{\ell} = \sum_{i=1}^{n} \ell_i^t/n$ be the average load and let $b = (\bar{\ell}, \ldots, \bar{\ell})$ be the *balanced distribution*. For each $t \geq 0$, define the *error* $e^{(t)}$ to be $\ell^{(t)} - b$. Let $-1 \leq \mu_1 \leq \mu_2 \leq \ldots \leq \mu_n = 1$ be the set of eigenvalues of M and denote $\gamma = \max_{\mu_i \neq 1} \{|\mu_i|\}$ to be the second largest eigenvalue of M. Let $||e^{(t)}||_2$ be the ℓ_2 norm of the error vector $e^{(t)}$. It can be shown that $||e^{(t+1)}||_2 = ||M \cdot e^{(t)}||_2 \leq \gamma \cdot ||e^{(t)}||_2$, which implies

$$||e^{(t)}||_2 \leq \gamma^t \cdot ||e^{(0)}||_2. \tag{2.1}$$

Subramanian and Scherson [107] observe similar relations between convergence time and the properties of the underlying network. From Equation (2.1), they obtain the following bound on the convergence time T:

$$\Omega\left(\frac{\log \sigma}{\Gamma}\right) \leq T \leq O\left(\frac{n\sigma}{\Gamma}\right)$$

and

$$\Omega\left(\frac{\log \sigma}{\Lambda}\right) \leq T \leq O\left(\frac{\sigma}{\Lambda^2}\right),$$

where n is the size of the network, σ is the standard deviation of the initial load distribution. Γ and Λ are the network's electrical and fluid conductance.

Muthukrishnan, Ghosh and Schultz [94] refer to the above diffusion model as the *first order scheme* and further generalize it to the so called *second order scheme*, where

$$L^t = \beta \cdot ML^{t-1} + (1 - \beta) \cdot L^{t-2},$$

with $0 \leq \beta \leq 1$ a constant. L^t relates not only to L^{t-1} but also to L^{t-2}, hence the name second order. They show that the second order scheme converges much faster than the first order scheme for suitably chosen values of β. Diekmann, Frommer and Monien [54] extend the idea of [94] and propose a general framework to analyze the convergence behavior of a wide range of diffusion-type approaches. They introduce the so called *Optimal Polynomial Scheme (OPS)*, which can determine an optimal balancing flow within m steps, where m is the number of distinct eigenvalues of the graph.

In [58] Elsässer, Monien and Schamberger analyze the diffusion algorithm for dynamically changing networks. The results are stated in Theorem 2.5.1. The proof technique is similar to the one in [54].

Dimension Exchange

In [46], Cybenko also investigates the following dimension exchange approach on a hypercube topology. In each round, a single dimension is taken and for every edge on that dimension, the algorithm equalizes the workloads of the nodes on both sides of the edge. It is shown that a sweep of all dimensions can actually balance the load globally; in fact, after d sweeps the system potential would be about e^{-2} ($\approx 1/8$) of the initial potential, d is the dimension of the hypercube.

In [63], Ghosh and Muthukrishnan study the dimension exchange approach for an arbitrary network G. To avoid concurrent load balancing actions they randomly generate a matching M_t in every round t. The nodes of the matching are then allowed to balance their load by exchanging half the load difference between every pair. Their proof uses a standard potential function argument. They first show that the probability for an edge to be included in the matching M_t is at least $1/8\delta$, where δ is the maximum degree of the network. Next, they estimate the expected potential drop by summing over all edges. They show that in each round the expected drop of ϕ is at least $\lambda_2/16\delta$. Here, λ_2 is defined as the second smallest eigenvalue of the *Laplacian matrix* of G. The Laplacian matrix of G is defined as $L = D - A$, with A denoting the adjacency matrix of G and $D = (d_{ij})$ with $d_{ij} = 0$ if $i \neq j$ and d_{ii} the degree of node i.

2.2.2 Discrete Load Balancing

Discrete load balancing, in which only integer tasks are allowed to be transferred, is a more realistic model than continuous load balancing. In this case the system can not be completely balanced. To see that consider the line graph as a network where the load of node i is simply i. The load is certainly not totally balanced but no neighbouring pair of nodes would balance their load. Unfortunately, discrete load balancing can not be analyzed using the algebraic technique of [46].

Quite often, the continuous model is used to bound the convergence time of discrete load balancing. Since the approximation error is mainly caused by rounding, it is not significant when the system is far from the balanced state (see [63, 94]). For the discrete version of their random matching based algorithm, by carefully calculating how much error can be introduced by rounding, Ghosh and Muthukrishnan [63] prove that, as long as $\phi \geq 2\delta n/\lambda_2$, the rounding can at most slow down the convergence time by a factor of two.

Besides, using the same rounding technique as above, Muthukrishnan, Ghosh and Schultz [94] show that in the case of the discrete version of their first order scheme, the initial potential φ_0 can be reduced to $O(\delta^2 n^2/\epsilon^2)$ in $O\left(\log \varphi_0/(1-(1+\epsilon)\gamma^2)\right)$ steps.

Rabani, Sinclair and Wanka [98] propose a more general technique to study the discrete load balancing. Their idea is to approximate the discrete system by idealized Markov chains. Let M be the diffusion matrix of a diffusion algorithm, and let $\gamma, \mu = 1 - |\gamma|$ be the second largest eigenvalue and the eigenvalue gap of M, respectively. Furthermore, let $K = \max_{i,j}\{|\ell_i - \ell_j|\}$ be the discrepancy of the initial load vector ℓ. They show for the idealized Markov chain that

$$\frac{2}{\mu} \ln\left(\frac{Kn^2}{x}\right) \tag{2.2}$$

rounds are sufficient to reduce the discrepancy to x. Next, to quantify the deviation of the actual load and the distribution generated by the Markov chain, they propose to use a natural quantity, the *local divergence* Ψ, which is the sum of load differences of the two systems across all edges of the network, aggregating over time. They obtain the following bound for Ψ: $\Psi(M) = O(\delta \log N/\mu)$. Finally, applying the knowledge of the second largest eigenvalue and combining this with Equation 2.2, they get fairly tight convergence results for various network topologies, e.g., line graph, de Bruijn network, degree-d expander etc.

In [57], Elsässer and Monien show that after applying the first order scheme for $k = \Theta(d \log(Kn)/\lambda_2)$ steps, the error in ℓ_2 norm $||e^{(k)}||_2$ can be reduced to $O(nd^2/\lambda_2)$, where K be the initial discrepancy (defined as above). Using a Markov chain based approach, Elsässer and Monien [57] propose a new discrete diffusion scheme which is fully randomized and distributed. Let δ be the maximum degree of the underlying graph. They show that, after $O\left(\frac{\delta}{\lambda_2}(\log n \log\log n + \log K)\right)$ steps, the algorithm can reduce the error bound $||e^{(k)}||_2$ to $O(\sqrt{n})$.

All the results above assume a *fixed* network. Recently, Elsässer, Monien and Schamberger [58] generalize the diffusion scheme to allowing dynamic networks. Let A_k be the average value of the ratio between the second smallest eigenvalue and the maximum degree during the first K iterations. They propose a diffusion algorithm that needs at most K steps to reduce the system potential from $\Phi(L)$ to $\epsilon\Phi(L)$, where $K = O(\ln(1/\epsilon)/A_K)$.

2.3 Model and New Results

We have n identical nodes that are connected by a network with maximum degree δ. Nodes are allowed to communicate with each other only if they are connected by an edge. Initially, each node stores some number of tasks. The total number of tasks is time-invariant, i.e., neither do new tasks appear, nor do existing ones disappear. The objective is to distribute the tasks as evenly as possible among the nodes whilst minimizing the number of load balancing steps.

The *load* of a node at time t is the number of tasks the node stores at that time. At each time step, every node compares its current load with the load of a subset of its neighbours, possibly with all of them. If the load of the node exceeds the load of such a neighbour by a certain amount, then it sends a certain number of tasks to that neighbour. Clearly, it will take a "long" time until the system is balanced if the number of tasks sent is "small" compared to the load difference. On the other hand, if this amount is too big, then load might bounce back and forth. To prevent that, the amount of load that a node is allowed to forward to a neighbour is typically upper bounded by a function of the difference d and the maximum degree, δ, e.g., $d/(\delta + 1)$.

Our main contribution is a new proof technique which can be used to analyze many diffusion-type load balancing algorithms, where the concurrent load balancing actions are the main challenge to the analysis. We demonstrate that our approach can be used to analyze the (continuous and discrete) diffusive load balancing in a variety of underlying network models.

The key idea is to first sequentialize the concurrent actions in a diffusion algorithm, and then study how much the concurrency can degrade the algorithm performance. We show that under certain conditions, the potential drops of both the sequentialized system and the concurrent system differ by a constant factor only. Hence, one can simply "neglect" the concurrency, and the remaining analysis can be easily done using existing techniques like the one in [63]. To illustrate how the idea works, we first analyze Algorithm 1, a classic diffusion algorithm similar to the ones studied in [46, 107, 94]. Next, we consider Algorithm 2, which allows every node to randomly find its balancing partner. We again analyze it using the same proof idea; this shows that our technique is quite general.

Specifically, Section 2.4 analyzes a diffusion algorithm (Algorithm 1) with concurrent load balancing actions. For the proof, we use a standard potential function Φ (similar to

the ones defined in [46, 63, 94, 107]). We can show that at each step, the potential drop of Algorithm 1 is at least some constant (0.5) times that of the corresponding sequentialized algorithm. In other words, the concurrency can degrade our algorithm performance by at most a factor of two. Finally, we adopt the proof idea in [63] to analyze the sequentialized algorithm so as to obtain the main convergence result (Theorem 2.4.4) for Algorithm 1.

Note that most existing results for diffusion algorithms consider the corresponding diffusion matrix of the network (see [31, 46, 107, 94]), while our result is expressed in terms of network parameters (e.g., the second-smallest eigenvalue of the Laplacian matrix, the maximum degree). Moreover, our approach is much simpler. Furthermore, due to the concurrent load balancing actions, our algorithm converges a constant times faster than the dimension exchange algorithm in [63].

Next, we analyze the discrete version of Algorithm 1 and obtain similar results to the ones in [63, 94]. We prove that as long as the potential is larger than a certain threshold (i.e., the system is "far" from the well-balanced state), the discrete case has similar convergence behavior to the continuous case. For the same discrete diffusion algorithm, our result (Theorem 2.4.6) is stronger than the one in [94], as it only requires the potential to be larger than a term linear in n instead of quadratic. Furthermore, compared to the discrete dimension exchange algorithm in [63], our algorithm is still a constant times faster.

In Section 2.5 we use our proof technique to get similar results to the ones in [58] for a dynamic network model where the active edges can change from round to round. In contrast to [58], we get also results for the discrete load balancing model.

In Section 2.6, we analyze Algorithm 2, which allows nodes to randomly choose balancing partners. Note that Algorithm 2 also contains concurrent load balancing actions since a node may have been chosen by many other nodes. Using the same proof idea to handle the concurrency, one can show that Algorithm 2 also converges quickly, as in each round the system potential drops by at least a constant factor in expectation. This implies that Algorithm 2 has a strict logarithmic convergence time which does not rely on any network parameters. Note that our results for this model are stronger than the ones that we would get by simply applying our results for the dynamic model. To our best knowledge this is the first time that the diffusion scheme is analyzed in a model where nodes are allowed to randomly choose balancing partners.

2.4 Diffusion on Fixed Networks

In this section we present our results in the standard diffusion model for arbitrary networks. Section 2.4.1 deals with the *continuous case*, where tasks can be arbitrarily split. In section 2.4.2 we show how to use our technique to obtain results for the discrete case.

2.4.1 Continuous Case

First we need some notation. $G = (V, E)$ is the underlying network. Let $\{e_1, e_2, \ldots e_{|E|}\}$ be the set of edges of G. For each node $i \in V$, let d_i be the degree of i, and let $\delta = \max_{i \in V} d_i$. $\alpha = \min_{S \subset V} \frac{|E(S,\overline{S})|}{\min(|S|,|\overline{S}|)}$ is the *edge expansion* of G, with $\overline{S} = V/S$, and $E(S, \overline{S})$ the set of edges with one endpoint in S and the other endpoint in $\overline{S}$. Furthermore, let $N(i) = \{j \in V | (i,j) \in E\}$ denote the set of all neighbours of node i. Let ℓ_i^t be the load of node i at the end of Round t. Whenever clear from the context we will simply write ℓ_i in the following. Then the vector $L = \{\ell_1, \ldots, \ell_n\}$ represents the entire load distribution. Now we are ready to define our load balancing algorithm.

Algorithm 1 *The diffusion algorithm on graph G*

1: **for** every node $i \in V$ in parallel **do**
2: **for** any $j \in N(i)$ **do**
3: **if** $\ell_i > \ell_j$ **then**
4: send $\frac{\ell_i - \ell_j}{4 \max\{d_i, d_j\}}$ load from node i to j

Similar to the result in [63], Theorem 2.4.4 (presented below) is a function of the edge expansion value and the maximum degree of G. Let $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of the Laplacian matrix of G (for the definition of Laplacian matrix, see Section 2.2.1). Let $L^t = \{\ell_1^t, \ldots, \ell_n^t\}, t \geq 0$ be the load vector after t balancing steps and $\overline{\ell} = \sum_{i=1}^n \ell_i / n$ the average load. In the following we will assume that all load vectors are *normalized*, i.e., $\ell_1^t \leq \ell_2^t \leq \ldots \leq \ell_n^t$. To analyze the algorithm, we will use the following potential function

$$\Phi(L^t) = \sum_{i=1}^{n} (\ell_i^t - \overline{\ell})^2.$$

Hence, $\Phi(L^{t-1}) - \Phi(L^t)$ is the potential drop in Round t.

We assign a weight $w_{ij} = \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4 \max\{d_i, d_j\}}$ to each edge $e = (i,j)$ in every round. The weight w_{ij} is the load that will be transferred over $e = (i,j)$ in Round t. Let $E^t = \{e_1^t, e_2^t, \ldots e_{|E|}^t\}$

be the set of edges sorted in increasing order of their weights. For the sake of the analysis, we now assume the edges are activated one by one starting with the edge e_1^t with the smallest weight. Then we can define $L^{t,k} = (\ell_1^{t,k}, \ldots, \ell_n^{t,k})$ to be the load vector right after the activation of the first k edges $e_1^t, \ldots e_k^t$ in Round t (applied to the load distribution L^{t-1}). $\Delta\Phi_\ell^t$ is the potential drop due to the activation of edge e_ℓ in Round t. Since L^t can be obtained from L^{t-1} by activating all edges in E one after another, we have $\Phi(L^{t-1}) - \Phi(L^t) = \sum_{e_\ell=(i,j)\in E} \Delta\Phi_\ell^t$. The next lemma gives a lower bound for the potential drop due to a single edge activation.

Lemma 2.4.1 *Fix a round t. For all edges $e_\ell = (i,j) \in E$ we have*

$$\Delta\Phi_\ell^t \geq w_{ij}|\ell_i^{t-1} - \ell_j^{t-1}|.$$

Proof. Assume $\ell_i^t \geq \ell_j^t$. Since all edges are activated in increasing order of their weights, the amount of load that node i can send to any other neighbour in Round t before the activation of e_k, is at most

$$w_{ij} = \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i, d_j\}}.$$

node i has at most $d_i - 1$ additional neighbours, hence it can send at most $(d_i - 1) \cdot \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i,d_j\}}$ load to other neighbours before the activation of edge (i,j). Consequently,

$$\begin{aligned}
\ell_i^{t,k-1} &\geq \ell_i^{t-1} - (d_i - 1) \cdot w_{ij} \\
&= \ell_i^{t-1} - d_i \cdot \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i, d_j\}} \right) + w_{ij} \\
&\geq \ell_i^{t-1} - \frac{1}{4}|\ell_i^{t-1} - \ell_j^{t-1}| + w_{ij}. \qquad (2.3)
\end{aligned}$$

Similarly, node j receives at most $(d_j - 1) \cdot \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i,d_j\}}$ before the activation of edge (i,j). Hence,

$$\begin{aligned}
\ell_j^{t,k-1} &\leq \ell_j^{t-1} + (d_j - 1) \cdot w_{ij} \\
&= \ell_j^{t-1} + d_j \cdot \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i, d_j\}} \right) - w_{ij} \\
&\leq \ell_j^{t-1} + \frac{1}{4}|\ell_i^{t-1} - \ell_j^{t-1}| - w_{ij}. \qquad (2.4)
\end{aligned}$$

Consequently,

$$\begin{aligned}
\Delta\Phi_\ell^t &= (\ell_i^{t,k-1} - \overline{\ell})^2 + (\ell_j^{t,k-1} - \overline{\ell})^2 - (\ell_i^{t,k} - \overline{\ell})^2 - (\ell_j^{t,k} - \overline{\ell})^2 \\
&= (\ell_i^{t,k-1})^2 + (\ell_j^{t,k-1})^2 - (\ell_i^{t,k})^2 - (\ell_j^{t,k})^2 \\
&= (\ell_i^{t,k-1})^2 + (\ell_j^{t,k-1})^2 - (\ell_i^{t,k-1} - w_{ij})^2 - (\ell_j^{t,k-1} + w_{ij})^2 \\
&= 2w_{ij}(\ell_i^{t,k-1} - \ell_j^{t,k-1} - w_{ij}) \\
&\geq 2w_{ij}\left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{2} + w_{ij}\right) \\
&\geq w_{ij}|\ell_i^{t-1} - \ell_j^{t-1}|.
\end{aligned}$$

The second equation is due to $\ell_i^{t,k-1} + \ell_j^{t,k-1} = \ell_i^{t,k} + \ell_j^{t,k}$. The first inequality is due to Inequalities 2.3 and 2.4. □

Now it is straightforward to lower bound the potential decrease in a whole round.

Lemma 2.4.2 $\Phi(L^{t-1}) - \Phi(L^t) \geq \frac{1}{4\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2$.

Proof.

$$\begin{aligned}
\Phi(L^{t-1}) - \Phi(L^t) &= \sum_{e_\ell=(i,j)\in E} \Delta\Phi_\ell^t \\
&\geq \sum_{(i,j)\in E} w_{ij}|\ell_i^{t-1} - \ell_j^{t-1}| \quad \text{(By Lemma 2.4.1)} \\
&= \sum_{(i,j)\in E} \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i, d_j\}} \cdot |\ell_i^{t-1} - \ell_j^{t-1}|\right) \\
&\geq \frac{1}{4\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2.
\end{aligned}$$

□

We shall use the following lemma from [63].

Lemma 2.4.3 *(Fact 2 from [63].)*

Let $\mathcal{L}$ be the Laplacian Matrix of our network, x' be the transpose of vector x and $v_1 = (1, 1, \ldots, 1)'$. We have

$$\lambda_2 = \min_x \left(\frac{x'\mathcal{L}x}{x'x} \quad | \quad x \perp v_1, x \neq 0\right),$$

where $x \perp v_1$ means that x is orthogonal to v_1.

Proof. Application of the the Courant-Fischer Minimax Theorem, see [63] for the full proof. □

It is now easy to derive the following theorem.

Theorem 2.4.4 *For any $\epsilon > 0$, after $T = \frac{4\delta \ln(1/\epsilon)}{\lambda_2}$ steps, we have $\Phi(L^T) \leq \epsilon \cdot \Phi(L)$.*

Proof. Fix a round t. First we lower bound $\frac{\Phi(L^{t-1})-\Phi(L^t)}{\Phi(L^t)}$. The idea is similar to [63]. Define x to be a vector of length n with $x_i = \ell_i^{t-1} - \bar{\ell}$. Note that $\sum_{i=1}^n x_i = 0$, and that x is orthogonal to $v_1 = (1, 1, \ldots, 1)^T$. Hence,

$$\begin{aligned}
\frac{\Phi(L^{t-1}) - \Phi(L^t)}{\Phi(L^{t-1})} &\geq \frac{\sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2}{4\delta \cdot \sum_{i=1}^n x_i^2} \qquad \text{(By Lemma 2.4.2)} \\
&= \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{4\delta \cdot \sum_{i=1}^n x_i^2} \\
&= \frac{1}{4\delta}\left(\frac{x'\mathcal{L}x}{x'x} \;\middle|\; \sum_{i=1}^n x_i = 0, x \neq 0\right) \\
&\geq \frac{1}{4\delta} \min_x \left(\frac{x'\mathcal{L}x}{x'x} \;\middle|\; x \perp v_1, x \neq 0\right) \\
&= \frac{\lambda_2}{4\delta}. \qquad \text{(By Lemma 2.4.3)} \qquad (2.5)
\end{aligned}$$

Hence, the potential drops by a constant factor in every round and we obtain

$$\Phi(L^T) \leq \left(1 - \frac{\lambda_2}{4\delta}\right)^T \Phi(L^0) = \left(\left(1 - \frac{\lambda_2}{4\delta}\right)^{\frac{4\delta}{\lambda_2}}\right)^{\ln(1/\epsilon)} \cdot \Phi(L) < \left(\frac{1}{e}\right)^{\ln(1/\epsilon)} \Phi(L) = \epsilon \cdot \Phi(L),$$

where the second inequality is due to $\forall 0 < a < 1$, $(1-a)^{1/a} < 1/e$. □

2.4.2 Discrete Case

In this section we analyze the discrete version of Algorithm 1 under the assumption that only integral amounts of tasks can be transferred. This means that for each edge (i, j), we transfer $\left\lfloor \frac{|\ell_i - \ell_j|}{4\max\{d_i, d_j\}} \right\rfloor$ tasks. Theorem 2.4.6 gives an upper bound for the balancing time for the discrete process. Note that it is no longer possible to balance the workload completely. Compared to the continuous version of the protocol, it takes longer for the discrete protocol to converge against a "nearly balanced state", but the difference is only a multiplicative constant.

Lemma 2.4.5 *Fix a round t. If $\Phi(L^{t-1}) \geq \frac{64\delta^3 n}{\lambda_2}$, $\frac{\Phi(L^t)-\Phi(L^{t-1})}{\Phi(L^{t-1})} \geq \frac{\lambda_2}{8\delta}$.*

Proof.

$$\begin{aligned}
\frac{\Phi(L^{t-1}) - \Phi(L^t)}{\Phi(L^{t-1})} &\geq \sum_{(i,j)\in E} \left\lfloor \frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i, d_j\}} \right\rfloor \cdot |\ell_i^{t-1} - \ell_j^{t-1}| \\
&\geq \sum_{(i,j)\in E} \left(\frac{|\ell_i^{t-1} - \ell_j^{t-1}|}{4\max\{d_i, d_j\}} - 1 \right) \cdot |\ell_I^{t-1} - \ell_j^{t-1}| \\
&\geq \frac{1}{4\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2 - \sum_{(i,j)\in E} |\ell_i^{t-1} - \ell_j^{t-1}| \\
&\geq \frac{1}{4\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2 - \sqrt{|E| \cdot \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2} \\
&\geq \frac{1}{4\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2 - \sqrt{n\delta \cdot \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2} \\
&\geq \frac{1}{8\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2.
\end{aligned}$$

The first inequality is due to Lemma 2.4.2. The fourth inequality follows from $\sum_{i=1}^m a_i \leq \sqrt{m \sum_i a_i^2}$. To show the last inequality, by Lemma 2.4.3, we get $\sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2 \geq \lambda_2 \Phi(L^{t-1}) \geq 64 n\delta^3$, hence $\sqrt{n\delta \cdot \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2} \leq \frac{1}{8\delta} \sum_{(i,j)\in E} (\ell_i^{t-1} - \ell_j^{t-1})^2$. □

Theorem 2.4.6 *After $T = \frac{8\delta \ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)}{\lambda_2}$ steps, $\Phi(L^T) < \frac{64\delta^3 n}{\lambda_2}$.*

Proof. By Lemma 2.4.5, in each round the potential drops by a constant factor as long as $\Phi(L^T) \geq \frac{64\delta^3 n}{\lambda_2}$. Hence, after $T = \frac{8\delta \ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)}{\lambda_2}$ steps, we have

$$\begin{aligned}
\Phi(L^T) &\leq \left(1 - \frac{\lambda_2}{8\delta}\right)^T \Phi(L^0) = \left(\left(1 - \frac{\lambda_2}{8\delta}\right)^{\frac{8\delta}{\lambda_2}} \right)^{\ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)} \cdot \Phi(L) \\
&\leq \left(\frac{1}{e}\right)^{\ln\left(\frac{\lambda_2 \Phi(L)}{64\delta^3 n}\right)} \cdot \Phi(L) = \frac{64\delta^3 n}{\lambda_2},
\end{aligned}$$

again, the second inequality is due to $\forall 0 < x < 1$, $(1-x)^{1/x} < 1/e$. □

Remark. Theorem 2.4.6 is stronger than Theorem 4 of [95] (see below), since we only require the potential to be linear in n, while Theorem 4 of [95] requires the potential to be at least

quadratic in n. Moreover, it is interesting to compare Theorem 2.4.6 with Theorem 4 in [57]. Theorem 2.4.6 indicates that the system potential $\Phi(L^T)$ can be reduced to $O(\delta^3 n/\lambda_2)$ after T steps. This is somewhat weaker than Theorem 4 in [57] which shows that the error bound $||e^T||_2$ can be reduced to $O(\sqrt{n})$ (Note that by definition $\Phi(L^T) = (||e^T||_2)^2)$. Yet, the convergence time bound of Theorem 2.4.6 is smaller than which of Theorem 4 in [57] (Note that $\log(\Phi(L)) = O(\log K + \log n)$ with K defined below). Moreover, Theorem 2.4.6 is for the general first order scheme (FOS) while Theorem 4 in [57] is for a specific algorithm that belongs to the second order scheme (SOS).

Theorem 2.4.7 *(Theorem 4 from [94].) For any $\epsilon < 1$, the discrete first order scheme reduces the potential to $O\left(\frac{d^2n^2}{\epsilon^2}\right)$ in $O\left(\frac{\log \Phi(L^0)}{1-(1+\epsilon)\gamma^2}\right)$ steps, where γ is the second largest eigenvalue of the network.*

Theorem 2.4.8 *(Theorem 4 from [57].) Let $G = (V, E)$ be a graph, let δ be the maximum vertex degree in G and let λ_2 be the second smallest eigenvalue of the Laplacian of G. Furthermore, let ω^0 be the initial load on G and $K = \max_i \{w_i^0 - \overline{w_i}\}$ the maximum load imbalance. If $\overline{\omega_i} \geq 32(n+1)\ln n$, then the randomized algorithm reduces the error $||e^T||_2$ to $O(\sqrt{n})$ with probability $1 - o(1/n)$ in*

$$T = O\left(\frac{d}{\lambda_2}(\log K + \log n \log\log n)\right)$$

iteration steps.

2.5 Diffusion on Dynamic Networks

In [58], Elsässer, Monien and Schamberger consider the diffusion process on *dynamic* networks, in which the set of nodes is fixed, but the set of of communication edges may vary from round to round. They assume that every node knows the edges that are active in a certain time step. The network can now be described by a sequence of "standard" graphs $(G_k)_{k\geq 0}$, where G_k is the underlying network at time step k. In this section we show how to use our analysis approach to get results for their network model. Similar to Section 2.4, we differentiate the *continuous* and the *discrete* cases.

2.5.1 Continuous Case

For the continuous case, Elsässer, Monien and Schamberger prove the following theorem. We can show exactly the same result for Algorithm 1 with our proof technique. In fact, Theorem 2.5.1 can be easily derived by Theorem 2.4.4.

Theorem 2.5.1 *(Theorem 1 from [58]). Denote $\lambda_2^{(k)}$ and $\delta^{(k)}$ to be the second smallest eigenvalue and the maximum degree of G_k respectively. Let $A_K = \frac{\sum_{k=1}^{K}\left(\lambda_2^{(k)}/\delta^{(k)}\right)}{K}$ be the average value of $\lambda_2^{(k)}/\delta^{(k)}$ occurring during the first K iterations. Algorithm 1 needs at most K steps to reduce the system potential from $\Phi(L)$ to $\epsilon\Phi(L)$, where $K = O\left(\frac{\ln(1/\epsilon)}{A_K}\right)$.*

Proof. Let L^K be the load vector after K rounds of applying the discrete version of Algorithm 1 on $(G_k)_{k\geq 0}$. Recall that $A_K = \frac{\sum_{k=1}^{K}\left(\lambda_2^{(k)}/\delta^{(k)}\right)}{K}$. Then, for $K \geq \frac{4\ln(1/\epsilon)}{A_K}$, it is true that

$$\begin{aligned}\Phi\left(L^K\right)/\Phi(L) &\leq \prod_{k=1}^{K}\left(1-\frac{\lambda_2^{(k)}}{4\delta^{(k)}}\right) < \prod_{k=1}^{K}\left(e^{-\lambda_2^{(k)}/4\delta^{(k)}}\right)\\ &= e^{-\sum_{k=1}^{K}\left(\lambda_2^{(k)}/4\delta^{(k)}\right)} = e^{-KA_K/4} \leq e^{-\ln(1/\epsilon)} = \epsilon.\end{aligned}$$

Here the first equation is due to Equation 2.5 of Theorem 2.4.4, the first inequality holds because $\forall 0 < x < 1$, $1 - x < e^{-x}$. □

2.5.2 Discrete Case

For the discrete case, we combine Theorem 2.5.1 and Lemma 2.4.5 and obtain the following theorem for the discrete version of Algorithm 1.

Theorem 2.5.2 *Let $\lambda_2^{(k)}, \delta^{(k)}, A_K$ be defined as above. The discrete version of Algorithm 1 needs at most K steps to reduce the system potential to*

$$\Phi^* = 64n \cdot \max_{k=1}^{K}\left\{\left(\delta^{(k)}\right)^3/\lambda_2^{(k)}\right\}, \text{ where } K = O\left(\frac{\ln\left(\Phi(L)/\Phi^*\right)}{A_K}\right).$$

Similar to Lemma 2.4.5, we can show that whenever the potential is larger than Φ^* defined above, the potential drops at least by a factor of $\lambda_2^{(k)}/(8\delta^{(k)})$ in iteration k. The rest proof is similar to Theorem 2.5.1, we omit the details.

2.6 Randomly Choosing Balancing Partners

In this section, we consider an alternative load balancing approach (Algorithm 2) which allows nodes to randomly choose their balancing partners. The algorithm proceeds in the following fashion: in each round, first every node randomly picks a balancing partner; later, load is transferred concurrently between the corresponding balancing partners. Note that unlike Algorithm 1, Algorithm 2 does not specify the underlying network topology. Using our analyzing technique to handle the concurrency, we can show that in each round, the system potential drops by at least a constant factor. This implies that Algorithm 2 has a strict logarithmic convergence time. Again, we first show results for the continuous case, and then for the discrete case.

2.6.1 Continuous Case

We denote by E the set of links whose endpoints are balancing partners, i.e., if node i chooses node j as balancing partner, we create a link (i, j) and add it to E. Moreover, let ℓ_i, $d(i)$ be the load and the number of balancing partners of node i. Our algorithm is as follows:

Algorithm 2 The diffusion algorithm that allows randomly picking balancing partners

1: $E = \emptyset$
2: **for** every node $i \in V$ do in parallel **do**
3: pick $j \in V$ uniformly at random
4: $E \leftarrow E \cup (i, j)$
5: **for** every node $i \in V$ do in parallel **do**
6: **for** every j such that $(i, j) \in E$ **do**
7: **if** $\ell_i > \ell_j$ **then**
8: send $\frac{\ell_i - \ell_j}{4 \max\{d_i, d_j\}}$ tasks from node i to j

Below we analyze Algorithm 2. First note that by the classic result of balls-into-bins games (see, for example, [13]), there is at least one vertex having $\Theta\left(\frac{\log n}{\log \log n}\right)$ balancing partners, with high probability. Consequently, one can not simply use the result in Section 2.4, which is in terms of the maximum degree of the underlying network. Instead, we prove the following result, which indicates that for a given link, it is unlikely for both sides of the link to have more than a constant number of balancing partners.

Lemma 2.6.1 *For a fixed link* $(i, j) \in E$, $\Pr[\max\{d_i, d_j\} \leq 5 \,|\, (i, j) \in E] > 0.5$.

Proof. By symmetry, we can assume that link (i,j) is built by node i. In this case, among the remaining $n-1$ nodes, there must be $d_i - 1$ nodes which choose i as their balancing partner. Since the probability for every node to choose i is $1/n$, we have $d_i \sim 1 + B(n-1, 1/n)$, where $B(n,p)$ is the binomial distribution. Next we consider node j. Note that node j has already connected to two links: (i,j) and another one that node j builds. Hence $d_j \sim 2 + B(n-2, 1/n)$ by similar reason as above. Next, we calculate $\Pr[d_i > 5 \,|\, (i,j) \in E]$.

$$\begin{aligned}
\Pr[d_i > 5 \,|\, (i,j) \in E] &= \Pr[B(n-1,1/n) > 4] = \Pr[B(n-1,1/n) \geq 5] \\
&\leq \binom{n-1}{5} \cdot \left(\frac{1}{n}\right)^5 < \left(\frac{ne}{5}\right)^5 \left(\frac{1}{n}\right)^5 \\
&= \left(\frac{e}{5}\right)^5 < 0.05.
\end{aligned}$$

Similarly, we can prove that $\Pr[d_j > 5 \,|\, (i,j) \in E] < \left(\frac{e}{4}\right)^4 < 0.25$. Using $\Pr[A \text{ or } B] \leq \Pr[A] + \Pr[B]$, the following holds.

$$\begin{aligned}
\Pr[\max\{d_i, d_j\} \leq 5 \,|\, (i,j) \in E] &= 1 - \Pr[d_i > 5 \text{ or } d_j > 5 \,|\, (i,j) \in E] \\
&> 1 - (\Pr[d_i > 5 \,|\, (i,j) \in E] + \Pr[d_j > 5 \,|\, (i,j) \in E]) \\
&> 1 - (0.05 + 0.25) > 0.5.
\end{aligned}$$

□

Before we prove Lemma 2.6.3, we show the following result indicating that the potential drop at some step t is a constant times of the current system potential.

Lemma 2.6.2 $\sum_{i=1}^{n} \sum_{j=1}^{n} (\ell_i^t - \ell_j^t)^2 = 2n \cdot \Phi(L^t)$.

Proof. Let $y_i = |\ell_i^t - \overline{\ell}|$, and denote A (or B) to be the set of indices i for which $\ell_i^t \leq \overline{\ell}$ (or $\ell_i^t > \overline{\ell}$ resp.). First observe that

$$\sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 = \sum_{i \in B} \sum_{j \in A} (y_i + y_j)^2, \tag{2.6}$$

and

$$\begin{aligned}
\sum_{i \in A} \sum_{j \in B} (y_i + y_j)^2 &= \sum_{i \in A} \sum_{j \in B} (y_i^2 + y_j^2 + 2 y_i y_j) \\
&= |B| \cdot \sum_{i \in A} y_i^2 + |A| \cdot \sum_{j \in B} y_j^2 + 2 \cdot \sum_{i \in A} y_i \cdot \sum_{j \in B} y_j.
\end{aligned} \tag{2.7}$$

Similar to (2.7), we get

$$\begin{aligned}\sum_{i\in A}\sum_{j\in A}(y_i-y_j)^2 &= \sum_{i\in A}\sum_{j\in A}\left(y_i^2+y_j^2-2y_iy_j\right)\\ &= 2\cdot|A|\cdot\sum_{i\in A}y_i^2-2\cdot\left(\sum_{i\in A}y_i\right)^2. \qquad (2.8)\end{aligned}$$

$$\begin{aligned}\sum_{i\in B}\sum_{j\in B}(y_i-y_j)^2 &= \sum_{i\in B}\sum_{j\in B}\left(y_i^2+y_j^2-2y_iy_j\right)\\ &= 2\cdot|B|\cdot\sum_{j\in B}y_j^2-2\cdot\left(\sum_{j\in B}y_j\right)^2. \qquad (2.9)\end{aligned}$$

By Equations 2.6, 2.7, 2.8 and 2.9, we have:

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n}(\ell_i^t-\ell_j^t)^2 &= \sum_{i\in A}\sum_{j\in B}(y_i+y_j)^2+\sum_{i\in B}\sum_{j\in A}(y_i+y_j)^2+\\ &\quad \sum_{i\in A}\sum_{j\in A}(y_i-y_j)^2+\sum_{i\in B}\sum_{j\in B}(y_i-y_j)^2\\ &= 2(|A|+|B|)\cdot\sum_{i\in A}y_i^2+2\left(|A|+|B|\right)\cdot\sum_{j\in B}y_j^2\\ &\quad +4\cdot\left(\sum_{i\in A}y_i\right)\cdot\left(\sum_{i\in B}y_j\right)-2\left(\sum_{i\in A}y_i\right)^2-2\left(\sum_{j\in B}y_j\right)^2\\ &= 2\left(|A|+|B|\right)\cdot\left(\sum_{i\in A}y_i^2+\sum_{j\in B}y_j^2\right)\\ &= 2n\cdot\Phi(L^t).\end{aligned}$$

Here the third equation holds since $\sum_{i\in A}y_i=\sum_{j\in B}y_j$. The last equation is true due to $|A|+|B|=n$ and $\Phi(L^t)=\sum_{i\in A}y_i^2+\sum_{j\in B}y_j^2$. □

Now we are ready to prove the following lemma.

Lemma 2.6.3 $E[\Phi(L^{t+1}\mid L^t=L)]\leq\frac{19}{20}\Phi(L)$.

Proof.

$$\begin{aligned}
E[\Phi(L^{t+1})|L^t = L] &= \Phi(L) - \sum_{i=1}^{n}\sum_{j=1}^{n}(\Pr[e_\ell = (i,j) \in E] \cdot \Delta\Phi_\ell(L)) \\
&\leq \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}|\ell_i - \ell_j| \\
&= \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{(\ell_i - \ell_j)^2}{4\max\{d_i, d_j\}} \\
&\leq \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\Pr[\max\{d_i, d_j\} \leq 5|(i,j) \in E] \cdot \frac{(\ell_i - \ell_j)^2}{4 \cdot 5}\right) \\
&\leq \Phi(L) - \frac{1}{40n}\sum_{i=1}^{n}\sum_{j=1}^{n}(\ell_i - \ell_j)^2 \\
&= \Phi(L) - \frac{\Phi(L)}{20} = \frac{19}{20}\Phi(L).
\end{aligned}$$

Here, the first inequality is from Lemma 2.4.1. The third equation is due to Lemma 2.6.1. The last equation holds by Lemma 2.6.2. □

Finally, we prove the following convergence theorem.

Theorem 2.6.4 *For $c > 0$, after $T \geq 120c\ln\Phi(L)$ rounds, $\Pr[\Phi(L^T) \leq e^{-c}] \geq 1 - \Phi(L)^{-c/4}$.*

Proof. For any $t > 0$, by linearity of expectation, we can iteratively use Lemma 2.6.3 to obtain

$$E\left[\Phi\left(L^{t+30}\right)\right] \leq \left(\frac{19}{20}\right)^{30}\Phi(L^t) \approx 0.21\Phi(L^t).$$

By Markov inequality, $\Pr[\Phi(L^{t+30}) < \Phi(L^t)/2] \geq 1/2$. Denote a stage to be 30 rounds. For any $c > 0$, let $k = 4\log\Phi(L)$. For stage $0 \leq i \leq k$, let X_i be a random variable defined as follows.

$$X_i = \begin{cases} 1 & \text{if } \Phi(L^{30(i+1)}) \leq \Phi(L^{30i})/2. \\ 0 & \text{otherwise.} \end{cases}$$

If $X_i = 1$, we say stage i is successful. Next we bound the number of successful stages. Let $X = \sum_{i=0}^{k} X_i$. Clearly $E[X] \geq k/2$. Applying Chernoff bound we get,

$$\Pr[X \leq c\ln\Phi(L)] \leq e^{-E[X]0.5^2/2} \leq e^{-k/16} \leq \Phi(L)^{-c/4}.$$

Hence, after $T = 30c \cdot k = 120c \cdot \ln \Phi(L)$ rounds, the number of successful stages is bigger than or equal to $c \ln \Phi(L)$ with probability at least $1 - \Phi(L)^{-c/4}$. Consequently, for $T \geq 120c \cdot \ln \Phi(L)$, $\Pr[\Phi(L^T) \leq e^{-c}]$ with probability at least $1 - \Phi(L)^{-c/4}$. □

Note that the random network model of this section can be viewed as a special case of the dynamic network model in Section 2.5. For random networks we are able to show that the potential decreases by a constant factor in each round. Theorem 2.5.1 does not give a constant factor drop for our random networks. This result is also related to the one of Ghosh and Muthukrishnan in [63], where they use random matchings for load balancing. To see the difference, first note that our random network may not be a matching. Second, our convergence result does not rely on any graph structure parameters while the result in [63] does.

2.6.2 Discrete Case

For the discrete case we use Algorithm 2 with one change. In every step, whenever $\ell_i > \ell_j$, we transfer $\left\lfloor \frac{\ell_i - \ell_j}{4 \max\{d_i, d_j\}} \right\rfloor$ tasks from ℓ_i to ℓ_j. We show the following result indicating that whenever the potential $\Phi(L)$ is bigger than a threshold of $3200n$, the potential decreases at least by a constant factor of $1/40$ in every iteration.

Lemma 2.6.5 *If* $\Phi(L) \geq 3200n$, $E[\Phi(L^{t+1} \mid L^t = L)] \leq \frac{39}{40}\Phi(L)$.

Proof. $E[\Phi(L^{t+1})|L^t = L]$

$$
\begin{aligned}
&= \Phi(L) - \sum_{i=1}^{n}\sum_{j=1}^{n} (\Pr[e_\ell = (i,j) \in E] \cdot \Delta\Phi_\ell(L)) \\
&\leq \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}|\ell_i - \ell_j| \\
&= \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} \left\lfloor \frac{(\ell_i - \ell_j)}{4\max\{d_i, d_j\}} \right\rfloor \cdot |\ell_i - \ell_j| \\
&\leq \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{(\ell_i - \ell_j)^2}{4\max\{d_i, d_j\}} + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} |\ell_i - \ell_j| \\
&\leq \Phi(L) - \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} \left(\Pr[\max\{d_i, d_j\} \leq 5 | (i,j) \in E] \cdot \frac{(\ell_i - \ell_j)^2}{4 \cdot 5} \right) + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} |\ell_i - \ell_j|
\end{aligned}
$$

$$
\begin{aligned}
&= \Phi(L) - \frac{1}{40n} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} (\ell_i - \ell_j)^2 + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} |\ell_i - \ell_j| \\
&\leq \Phi(L) - \frac{1}{40n} \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} (\ell_i - \ell_j)^2 + \frac{1}{n}\sqrt{n^2 \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} (\ell_i - \ell_j)^2} \\
&\leq \Phi(L) - \left(\frac{1}{40n} - \frac{1}{80n}\right) \cdot \sum_{i=1}^{n}\sum_{j=1}^{n} (\ell_i - \ell_j)^2 \\
&= \Phi(L) - \frac{\Phi(L)}{40} = \frac{39}{40}\Phi(L).
\end{aligned}
$$

Here, the first inequality is from Lemma 2.4.1. The third equation is due to Lemma 2.6.1. The fourth inequality holds by $\sum_{i=1}^{m} a_i \leq \sqrt{m \sum_i a_i^2}$. The last inequality is true since

$$\sum_{i=1}^{n}\sum_{j=1}^{n} (\ell_i - \ell_j)^2 = 2n\Phi(L) \geq 6400n^2.$$

The last equation follows from Lemma 2.6.2. □

Finally, similar to Theorem 2.6.4, we directly obtain the following theorem:

Theorem 2.6.6 *$\forall c > 0$, after $T \geq 240c \ln\left(\frac{\Phi(L)}{3200n}\right)$ rounds, $\Pr[\Phi(L^T) \leq 3200n] \geq 1 - \left(\frac{\Phi(L)}{3200n}\right)^{-c/4}$.*

2.7 Summary

We have proposed a new proof technique that can be used to analyze many parallel diffusive load balancing algorithms. The idea is to first sequentialize a diffusion algorithm with concurrent load balancing actions, and then to show that the potential decrease of the concurrent system is at most a constant factor compared to which of the corresponding sequential system. We have demonstrated the strength of the technique by analyzing continuous and discrete diffusive load balancing algorithms for both fixed network and randomly changing networks.

We believe that this simple idea is useful in the analysis of many distributed systems with concurrent actions. For instance, this idea is applied in the analysis of a parallel randomized load balancing protocol in Section 3.5. In the future, we can apply this idea to analyze load balancing on other distributed systems, for example, the ad hoc network. This

problem is somewhat similar to the problem of randomly choosing load balancing partners in Section 2.6. In particular, we can either assume that the topology of an ad hoc network is arbitrary, or belongs to some random network classes such as Erdös Rènyi model [55] or random geometric graphs [95].

Chapter 3

Weighted Balls-into-bins Games

In this chapter we study the balls-into-bins game, where we have a set of independent balls and the goal is to allocate them into a set of bins in a balanced manner. This problem has been used to model a large set of real life applications in distributed systems where balls and bins represent tasks and resources, respectively. We assume that every ball is associated with some *weight*, which can represent the resource requirement of tasks, i.e., memory or running time.

We consider two different scenarios, the *static sequential* game and the *selfish allocation* game. In the static sequential game, balls come sequentially and need to be placed in such order. We study how the outcome of the game, i.e., the *expected maximum load* of any bin, is influenced by the game parameters such as the *distribution* of ball weights, and the *order* that balls are allocated. In the selfish allocation game, all the balls have some initial locations and need to be reallocated to achieve load balancing. We analyze a natural, distributed reallocation algorithm. Our goal is to bound the convergence time of the algorithm, i.e., the number of rounds for the system to reach (or get close to) some equilibrium state.

3.1 Introduction

The balls-into-bins game, also referred to as occupancy problem or allocation process, is a well-known and much studied model in distributed computing. Generally speaking, the goal of balls-into-bins games is to allocate a set of independent balls into a set of bins so that the maximum number of balls in any bin is minimized. It is assumed that there is no global mechanism to realize this goal and all the balls are independent to each other. Furthermore,

every ball is allowed to collect partial load information by querying a small set of bins, but it is not allowed to communicate with other balls.

This well-defined mathematic problem has been used to model many practical distributed applications, e.g., dynamic resource allocation, distributed client server systems, network routing and hashing etc. [13, 68, 90]. For instance, we consider a client server system where each client issues some tasks which can be assigned to each server. In order to minimize the response time of tasks, we need to distribute these tasks to servers as evenly as possible. Note that the tasks do not have any information about other tasks in the system and they have to be allocated independently. In this case, the balls-into-bins game is an excellent model that allows us to effectively use the system. In those applications above, tasks are often heterogenous, for example, the tasks in the client server system can have different sizes. In this chapter, we investigate balls-into-bins games in the content of weighted balls. We assume that each ball (task) is associated with some positive weight representing its resource requirements, i.e., memory or running time.

Balls-into-bins games can be categorized in many ways. We can have either *sequential* or *parallel* games. In the sequential game, balls arrive one after another and have to be placed in such order. The most common approach is to choose d bins independently and uniformly at random and place the ball into the least loaded bin [68, 13, 20, 110]. In the parallel game, balls arrive in batches while balls in the same batch must be allocated concurrently. For the parallel game, the previous approach is no longer applicable and some kind of scheduling process is required. For more about the parallel game, see also [3, 106, 19]. Furthermore, we distinguish between both *static* and *dynamic* games. In the case of static games, the set of balls is fixed. None of the balls will be deleted and no new ball will ever arrive. In the dynamic case, we do not have a fixed number of balls but rather balls may arrive according to some arrival process and leave the system according to some deletion process. An arrival (or deletion) process specifies the points of time and the number of balls being injected into (or deleted from) the system. For example, in [87], Mitzenmacher considers a model that the balls arrive as a Poisson stream of rate λn, and each ball has service time that is exponentially distributed with mean one. See also [13, 48] for more examples.

In this chapter, we consider two different settings of balls-into-bins games, the *static sequential game* and the *selfish allocation game.*

Static Sequential Game In the first part of this chapter (Section 3.4), we study the static sequential game, where balls arrive one after another without initial locations and have to be allocated in such order. A well-known approach is to have every ball choose $d \geq 1$ bins independently uniformly at random and pick the bin with the lightest load. An advantage of this approach is that it does not need any global information (i.e., system load configuration) for allocating balls. Moreover, this approach causes very little overhead since each ball is only allowed to query a small number of bins for finding its destination. If every ball is only allowed to pick one bin (i.e., $d = 1$), the game is called *single-choice* balls-into-bins game, otherwise *multiple-choice*. As shown in [68, 13], the multiple-choice approach could result in a drastic (exponential) decrease of the maximum load over the single-choice approach. During the past years, there has been extensive study on this phenomenon and many significant results have been obtained. See Section 3.2 for a survey of both the single-choice and the multiple-choice approaches.

Most work done so far assumes that the balls are uniform and indistinguishable. We concentrate on the *weighted* case where every ball $i \in [m]$ is associated with a weight $w_i > 0$. Let the *load* of a bin denote the sum of the weights of the balls allocated to that bin. In [17] the authors study the weighted balls-into-bins game by comparing it with its corresponding uniform games. More specifically, they compare the maximum load of a game with m weighted balls with maximum weight of 1 and total weight $W = w_1 + \cdots + w_m$ to a game with approximately $4W$ uniform balls. They show that the maximum load of the weighted game is not larger than that in the uniform system. Their approach can be used for a variety of balls-into-bins games and can be regarded as a general framework.

However, the results of [17] seem to be somewhat unsatisfactory. The authors compare the allocation of a number of "light" weighted balls with an allocation of fewer "heavy" uniform balls. Intuitively, since the case with light balls comes with more random choices, the result should be better to allocate many light balls compared to fewer heavy balls. In light of this, we study how the weight distribution affects the results of both the single-choice and the multiple-choice games. We show that for the single-choice game, it is indeed true that allocating more balanced weight distribution is always better. To show this result we use the majorization technique similar to the one in [13]. Later, we prove that for the multiple-choice game, surprisingly, a more balanced weight distribution does NOT always imply a smaller expected maximum load.

Another related question is, how does the order of allocating balls affect the outcome

of the game? We note that in the single-choice game, since the random choices of all balls are independent, the order does not make any difference. Yet, the question becomes subtle for the multiple-choice game. Intuitively, it should be better to allocate the heavy balls first, then the light balls allocated later will tend to fill the "holes" left by the large ones. Hence, we ask, does the decreasing order always yields the minimum expected maximum load? Similarly, does the increasing order always yield the maximum expected maximum load?

Selfish Reallocation Game In the second part of this chapter (Section 3.5), we consider the problem of dynamically reallocating (or re-routing) m balls among a set of n bins. This problem can be used to model many practical applications in large scale networks, e.g. the internet, which are open structured and lack of central authorities to guide users' behaviors. Particularly, in these applications, users are *selfish* in that they only aim at minimizing their own costs without regard for the overall cost. We model the selfish users as balls, the resources (e.g., network links, processors etc) as bins.

We assume that initially each ball has chosen some bin. (Note that this is different to the static sequential game introduced above, where balls have to be placed one after another). Then we apply the following iterative, distributed algorithm to reallocate balls to bins for load balancing. In each step, every ball chooses one bin at random. It then compares the load of its current host bin with the load of the randomly chosen bin. If the load difference is above a certain threshold then the ball will migrate to the other bin with a probability that is proportional to the load difference of these two bins. In this algorithm, each ball acts selfishly in that it only tries to minimize the loads of its host bin. Furthermore, there is no central control in the system and all the migrations take place in parallel.

We express our results using the notion of Nash equilibrium and its variations. In general, a *Nash equilibrium* is a state in which users (balls) do not have an incentive to unilaterally change their current strategy. In our problem, the Nash equilibrium represents a state that none of the balls has an incentive to migrate to other bins. We shall also consider a notion of approximate equilibria, namely ϵ-Nash equilibria, which describes states where no user can benefit by more than ϵ if he/she unilaterally changes his/her current decision. Our major goal is to bound the *convergence time*, i.e., the number of time steps for the system to reach some Nash equilibrium (or ϵ-Nash equilibrium).

The rest of the chapter is organized as follows. Section 3.2 consists of some related work.

In Section 3.3 we introduce our model and summarize the new results. We study the static sequential game and the selfish allocation game in Section 3.4 and Section 3.5, respectively. Finally we conclude in Section 3.6.

3.2 Related Work

3.2.1 Static Sequential Game

We review related work for both the single-choice and multiple-choice balls-into-bins games.

Single-choice Game

For the single-choice balls-into-bins game, every ball randomly chooses a destination bin. It is well-known that (e.g., see [90]), if n balls are allocated into n bins using this strategy, the fullest bin has $\log n/\log\log n+O(1)$ balls with high probability [1] (w.h.p. or more accurately $\Gamma^{-1}(n)-\frac{3}{2}+o(1)$. More generally, to allocate $m \geq n \ln n$ balls into n bins, the maximum load is $(m/n)+\Theta(\sqrt{m\log n/n})$ w.h.p. [90, 20]. Note that the deviation term $(\Theta(\sqrt{m\log n/n}))$ grows linearly with $\sqrt{m}$. This is not satisfactory since in practice we often have $m \gg n$.

In [104], Sanders considers the single-choice game in a weighted setting. Assume that both the total weight of the balls W and the maximum ball weight $w_{\max}$ are fixed. Then the expected maximum load is maximized when $W/w_{\max}$ balls of weight $w_{\max}$ are allocated.

In [70], Koutsoupias, Mavronicolas and Spirakis consider the random allocation of weighted balls. Similar to [17], they compare the maximum load of an allocation of weighted balls to that of an allocation of a smaller number of uniform balls with a larger total weight. They repeatedly fuse the two smallest balls together to form one larger ball until the weights of all balls are within a factor of two of each other. They show that the bin loads after the allocation of the weighted balls are *majorized* by the loads of the bins after the allocation of the balls generated by the fusion process. Their approach also applies to more general games in which balls can be allocated into bins with non-uniform probabilities.

Multiple-choice Game

For the multiple-choice balls-into-bins game, every ball randomly chooses $d \geq 2$ bins uniformly at random and allocate itself into the one with the minimum load. During recent

[1] We say an event A occurs with high probability, if $\Pr[A] \geq 1 - 1/n^{\alpha}$ for some constant $\alpha \geq 1$.

years there has been much research on this problem, see [90] for a nice overview. The first rigorous analysis for the multiple-choice game is due to Karp, Luby and Meyer aur der Heide [68], which studies the use of two hash functions in the context of PRAMs (Parallel Random Access Machines). According to [90], there are three main techniques to analyze balls-into-bins games, *layered induction, witness trees* and *differential equation* techniques.

Layered Induction Technique Azar et al. [13] first introduce the layered induction technique to prove tight results for the case when $m = O(n)$. They show that after placing m balls the maximum load is $\Theta(m/n + \log\log n/\log d)$, w.h.p. [2] The idea is to bound the number of bins with at least $k+1$ balls by the number of bins with at least k balls. For example, assume there are n balls to be allocated into n bins. Let β_k be the upper bound for the number of bins with at least k balls w.h.p. Note that by pigeonhole principle, $\beta_6 \leq n/6$. Then using an induction approach one can then bound β_{i+6} in terms of β_{i+6-1}. Note that every stage the bound holds with high probability. The probability that there does exist a bin with a load of $k+6$ is bounded by the sum of fail probabilities in all proof stages. The authors then apply a similar inductive argument and stochastic domination to establish a matching lower bound.

In [20], Berenbrink et al. analyze Greedy[d] for $m \gg n$. The authors eventually tighten the upper bound result of the maximum load to $m/n + \log\log n + O(1)$, w.h.p. This shows that the multiple-choice process behaves inherently different from the single-choice process, where the difference between the maximum load and the average load depends on m. They first show a "short memory" property of the Greedy process, i.e., no matter what the initial situation is, after a polynomial number of additional balls the maximum load of any bin can again be bounded as expected. This memoryless property separates the multiple from the single-choice approaches, in that for any number of bins, the difference between the optimal and the multiple-choice allocation is bounded by a constant, instead of increasing polynomially with m. Hence, it is sufficient to consider the case when $m = poly(n)$. The rest proof also utilizes the layered induction technique but the idea is very different. In particular, they prove bounds not only for the balls lying above the average load, but also for those balls lying below average load.

[2] We say an event A happens with high probability (w.h.p.), if $\Pr[A] > 1 - n^{-1}$.

Witness Tree Technique The witness tree technique is first used to analyze balls-into-bins games by Meyer auf der Heide, Scheideler and Stemann [11] and is further exploited in [106]. The idea is similar to the analysis technique *delayed sequences* in the study of randomized routing algorithms [108, 6]. When applying to balls-into-bins games, the idea is to specify those events lying on the past that "witness" the occurrence of some heavily loaded bin. Assume that there is some bin v with at least k balls. Let b be the last ball allocated to v. There must be $d-1$ other bins with load at least $k-1$ since b is allowed to query d bins and place itself into the one with the smallest load. Similarly, for each bin with load $k-1$, there are $d-1$ bins with a load at least $k-2$, and so on. These events naturally form a so-called witness tree. Then, in order to bound the probability that some bin gets load at least k, we can turn to calculate the total probability of the occurrences of these witness trees. Vöcking [110] uses the witness tree technique to analyze a variant of the multiple balls-into-bins game, called *Always-Go-Left*, which introduces a new tie-breaking mechanism that always picks the leftmost bin instead of picking one arbitrarily. Surprisingly, the Always-Go-Left algorithm can achieve a maximum load of $m/n + \log\log n/d + O(1)$, w.h.p.

Differential Equation Technique The idea of differential equation technique is to study the corresponding *continuous* system of the (discrete) balls-into-bins game. It is well-known that in many cases, the continuous systems are easier to analyze by differential equations. Using this technique, Mitzenmacher, Prabhakar and Shah [89] show that a similar performance gain to the multiple-choice game can be achieved by introducing memory. More specifically, they show that if every ball only gets one random choice, and meanwhile it can also pick the least loaded bin after allocating the last ball, the maximum load is $\log\log n/(2\log\Phi) + \Theta(1)$ w.h.p. where $\Phi = (\sqrt{5}+1)/2$ is the golden ratio. For more about this technique, see e.g., [88, 87, 90].

3.2.2 Selfish Reallocation Game

Next we review previous work for the selfish reallocation game. In [59], Even-Dar, Kesselman and Mansour introduce the idea of using a potential function to measure the closeness between a system state and the balanced allocation. They use this idea to show convergence for sequences of randomly-selected "best response" moves in a more general setting in which balls may have variable weights and bins may have variable capacities. Best response means

that every time a ball(task) always picks the move that incurs the smallest cost for itself. Since they consider only the best-response moves, it is necessary for them to consider only strictly sequential algorithms.

Goldberg [64] considers a algorithm in which every ball select one alternative bin at random and migrate if the selected bin has lower load. The algorithm may be implemented in a weakly distributed sense, requiring that migration events take place one at a time, and costs are updated immediately. He proves an upper bound of $O(w_{max}^2 m^4 n^5 \log(mn))$ for the convergence time, where m is the number of balls, n is the number of bins, w_{max} is the ratio between the largest and the smallest ball weights. If all balls are of integer weights between 1 and w_{max}, the convergence time is $O(m^2 n w_{max})$. He also shows an $\Omega(n^2)$ lower bound for the convergence time.

Even-Dar and Mansour [60] allow concurrent, independent reallocation decisions where balls are allowed to migrate from bins with load above average to bins with load below average. They show that the system reaches a Nash equilibrium after expected $O(\log\log m + \log n)$ rounds. Their proof is also based on the standard potential function technique. However, their algorithm requires balls to know certain amount of global knowledge in order to make their decisions. A ball needs to know whether its bin is overloaded, i.e., with a load larger than average. The authors also show a logarithmic convergence rate for a wide range of rerouting strategies.

In [61], Fischer, Räcke, and Vöcking investigate convergence to Wardrop equilibria for both asymmetric and symmetric rerouting games. In asymmetric games, tasks may be associated with different latency functions. They consider a set of rerouting algorithms called *adaptive sampling*, where in each round, each ball samples an alternative routing path with its current latency. If the ball observes that it can improve its latency, it then switches with some probability depending on the improvement to the better path. The authors prove the first polynomial bounds on the convergence time of adaptive rerouting policies for classes of latency functions with bounded *relative slope*. A differentiable latency function ℓ has relative slope d if $\ell'(x) \leq d\ell(x)/x$ for all x in the entire range. The authors also show the necessity of adaptive sampling by proving an exponential lower bound result for the static sampling methods.

Chien and Sinclair [35] study the ability of a set of distributed, local algorithms to rapidly reach the approximate Nash equilibrium. They show that for the symmetry congestion game, if the latency function d_e of any edge (bin) e satisfies the so called "bounded jump"

condition, i.e., $d_e(k+1) \leq \alpha d_e(k)$ for all $k \geq 1$, the convergence to an ϵ-Nash equilibrium occurs within $\lceil n\alpha\epsilon^{-1}\log(nc)\rceil$ steps, n is the number of balls. They also show that it is necessary to consider the approximate Nash equilibrium, in that the problem of finding a (real) Nash equilibrium in symmetric congestion game satisfying the α-bounded jump condition with $\alpha = 2$ is PLS-Complete.

Berenbrink et al. [22] consider a strongly distributed system consisting of selfish users. They consider only uniform balls and uniform bins. They show an upper bound of $O(\log\log m + n^4)$ on the expected convergence time in their model as well as a lower bound of $\Omega(\log\log m + n)$. They furthermore derive bounds on the convergence time to an approximate Nash equilibrium as well as an exponential lower bound for a slight modification of their algorithm. In fact, the modification is possibly even more natural than the one with the polynomial upper bound in that it results in a perfectly balanced distribution in expectation after only one step whereas the previously mentioned algorithm does not have this property.

3.3 Model and New Results

We first introduce the model we are working on. We have m balls and n bins. Let $[m]$ denote $\{1,\ldots,m\}$. Every ball $i \in [m]$ is associated with some weight $w_i \geq 1$. Let $w = (w_1,\ldots,w_m)$ be the vector of ball weights and $W = \sum_{i=1}^{m} w_i$ be the total weight of the balls. If $w_1 = w_2 = \ldots = w_n$, we say the game is *uniform*. In this case, we normalize the ball weights such that $w_i = 1$ for all $i \in [m]$. The *load* of a bin is defined as the total weight of balls located in that bin. Then let $\bar{x} = W/n$ be the average load of all bins.

In the following, we summarize our new results for both the static sequential game and the selfish reallocation game.

3.3.1 Static Sequential Game

For the static sequential game, we allocate a set of weighted balls into bins in a sequential fashion. We consider the well-known approach that to have every ball pick $d \geq 1$ bin independently uniformly at random, and pick the one with the lightest load. We study how the weight distribution and the order in which we allocate balls influence the outcome (expected maximum load) of the game.

Section 3.4.2 studies the single-choice game. In Theorem 3.4.8 we fix the number of balls and show that the expected maximum load is smaller for more balanced ball weight

vectors. This also holds for the sum of the loads of the i largest bins. One could say that the *majorization is preserved*: if one weight vector majorizes another one, then we have the same order with respect to the resulting expected bin load vectors. Hence, the expected maximum load is minimized when we have uniform balls. To prove Theorem 3.4.8, we use an inductive approach. The idea is to use majorization together with T-transformations (see the definition in Section 3.4.1), which allow us to compare sets of balls that only differ in *one* pair of balls. Corollary 3.4.10 extends the results showing that the allocation of a large number of small balls with total weight W ends up with a smaller expected maximum load than the allocation of a smaller number of balls with the same total weight. We also show that the results are still true for many other random functions that are used to allocate the balls into the bins. Our results are stronger than the ones of [70] since we compare arbitrary weight distributions with the same total weight. Compared to [70] we also allow for the same number of balls. In addition, we consider the entire load distribution and not only the maximum load.

Section 3.4.3 deals with the multiple-choice game. The main result here is Theorem 3.4.17. It shows that, for sufficiently many balls, allocation of uniform balls is *not* necessarily better than allocation of weighted balls. It is better to allocate first the big balls and then some smaller balls on top of them, instead of allocating the same number of average sized balls. This result uses the memoryless property of [20]. For fewer balls we show in Theorem 3.4.18 that the majorization order is not generally preserved.

The previous results for the single-choice game use the majorization technique inductively. Unfortunately, it seems difficult to use T-transformations and the majorization technique to obtain results for weighted balls in the multiple-choice game. We also present several examples showing that, for the case of a small number of balls with multiple-choices, the maximum load is not necessarily smaller if we allocate more evenly weighted balls.

3.3.2 Selfish Reallocation Game

For the selfish reallocation game, initially every ball has chosen some bin. Then we apply some iterative, distributed reallocation algorithm to balance the work load. We study the convergence time of the algorithm, i.e., the number of rounds for the system to terminate.

In Section 3.5, we consider the weighted case where each ball $i \in [n]$ is associated with some positive weight w_i. We propose a greedy distributed reallocation algorithm (Algorithm 4) that is similar to the one in [22]. Theorem 3.5.1 shows that after $O(mn\Delta^3\epsilon^{-2})$ steps,

the system converges to the ϵ-Nash equilibrium with probability at least 4/5. To our best knowledge, this is the first attempt in such model to allows weighted balls. Our analysis is based on the potential function technique. The idea is to use an appropriate potential function to measure the distance between some system state with the equilibrium state. We then show that the system potential always decreases in expectation in one single step. Corollary 3.5.10 shows that if all balls are of integer weights, the convergence occurs within $O(mn\Delta^5)$ steps. In addition, we prove a lower bound of $\Omega(m\Delta/\epsilon)$ for the convergence time (Observation 3.5.11).

In Section 3.5.3, we apply our proof technique above to the uniform case where all the balls are identical. We show that our algorithm converges to the (real) Nash equilibrium in time $O(\log m + n \log n)$ steps w.h.p. This improves the previous result of $O(\log \log m + n^4)$ in [22] for small values of m. We also demonstrate that we can in fact combine our algorithm and the algorithm in [22] to obtain an $O(\log \log m + n \log n)$ convergence time w.h.p. Finally, we show a matching lower bound result (Observation 3.5.18).

3.4 Static Sequential Game

In this section we focus on the static sequential games, where a fixed number of balls, m, are allocated one after the other. A well-known approach is to let every ball choose $d \geq 1$ bins independently and uniformly at random, and allocate itself into the bin with minimum number of balls (ties are broken arbitrarily). In the following, we will refer to this algorithm as Greedy[d] similar to [13].

Algorithm 3 Algorithm Greedy[d]

for each ball b **do**
  Choose d bins $u_1, \ldots, u_d$ independently uniformly at random
  Place ball b into the bin with the least load among $u_1, \ldots, u_d$.

The status of an allocation is described by a *load vector* $L(w) = (\ell_1(w), \ldots, \ell_n(w))$, where ℓ_i is the load of the i-th bin after the allocation of a weight vector w. Whenever it is clear from the content we shall drop "w" and write instead $L = (\ell_1, \ldots, \ell_n)$. In some cases we consider the change that occurs in an allocation after allocating some number of additional balls. Then we define L_t to be the load vector after the allocation of the first t balls with weights $w_1, \ldots, w_t$ for $1 \leq t \leq m$. In many cases we will *normalize* a load vector

L by assuming a non-increasing order of bin loads, i.e. $\ell_1 \geq \ell_2 \geq \cdots \geq \ell_n$. We then define $S_i(w) = \sum_{j=1}^{i} \ell_j(w)$ as the total load of the i highest-loaded bins. Again, when the context is clear we shall drop the "w" and write $S_i = \sum_{j=1}^{i} \ell_j$. Finally, let $\Omega = [n]$. Before we proceed, we shall first introduce our major tool, the majorization technique [13, 86].

3.4.1 Majorization and T-transformations

To compare two load vectors and also the balancedness of vectors of ball weights, the concept of *majorization* is essential. We first give the definition of majorization (from [86]).

Definition For two normalized vectors $w = (w_1, \ldots, w_m) \in \mathbb{R}^m$ and $w' = (w'_1, \ldots, w'_m) \in \mathbb{R}^m$ with $\sum_{i=1}^{m} w_i = \sum_{i=1}^{m} w'_i$, we say that w' *majorizes* w, written $w' \succ w$, if $\sum_{i=1}^{k} w'_i \geq \sum_{i=1}^{k} w_i$ for all $1 \leq k \leq m$.

Majorization is a strict partial ordering between (normalized) vectors of the same dimensionality. Intuitively, vector v' *majorizes* another vector v if v is "more spread out", or "more balanced", than v'. In the following, if we refer to a weight vector w that is more balanced than weight vector w', we mean that w' majorizes w. We will use the term majorization if we refer to load vectors.

Some examples are:

$$(1, \underbrace{0, \ldots, 0}_{m-1}) \succ (1/2, 1/2, \underbrace{0, \ldots, 0}_{m-2}) \succ \ldots \succ (\underbrace{1/(m-1), \ldots, 1/(m-1)}_{m-1}, 0) \succ (\underbrace{1/m, \ldots, 1/m}_{m}).$$

For the sake of our analysis, we give a slightly different alternative definition of majorization as follows.

Majorization Let w and w' be two weight vectors with m balls, and let Ω^m be the set of all possible random choices for Greedy[d] applied on m balls. Define $w(\omega)$ (respectively, $w'(\omega)$) to be the allocation resulting from the choices $\omega \in \Omega^m$, and let $f : \Omega^m \to \Omega^m$ be a *one-to-one* correspondence. Then we say that w' is majorized by w if there exists a function f such that for any $\omega \in \Omega^m$ we have $w(\omega) \succ w'(f(\omega))$.

A slightly weaker form of the majorization is the *expected majorization* defined below. We will use it in order to compare the allocation of two different load vectors with each other.

Expected majorization Let w and w' be two weight vectors with m balls, and let Ω^m be the set of all possible random choices. Let $L(w,\omega) = (\ell_1(w,\omega),\ldots,\ell_n(w,\omega))$ (respectively, $L'(w',\omega) = (\ell_1(w',\omega),\ldots,\ell_n(w',\omega))$) be the normalized load vector that results from the allocation of w (respectively, w') using $\omega \in \Omega^m$. Let $S_i(w,\omega) = \sum_{j=1}^{i} \ell_j(w,\omega)$ and $S_i(w',\omega) = \sum_{j=1}^{i} \ell_j(w',\omega)$. Then we say that $L(w')$ is *expectedly majorized* by $L(w)$ if for all $i \in [n]$, we have $E[S_i(w)] \geq E[S_i(w')]$. (The expectation is over all possible n^m elements, selected uniformly at random, in Ω^m.)

Now we introduce a class of linear transformations on vectors called *T-transformations* which are crucial to our later analysis. We write

$$w \overset{T}{\Longrightarrow} w',$$

meaning that w' can be derived from w by applying one T-transformation. Recall that a square matrix $\Pi = (\pi_{ij})$ is said to be *doubly stochastic* if all $\pi_{ij} \geq 0$, and each row sum and column sum is one. Π is called a *permutation matrix* if each row and each column contains exactly one unit and all other entries are zero (in particular, a permutation matrix is doubly stochastic).

T-transformation A *T-transformation* matrix T has the form $T = \lambda I + (1-\lambda)Q$, where $0 \leq \lambda \leq 1$, I is the identity matrix, and Q is a permutation matrix that swaps exactly two coordinates. Thus, for some vector x of correct dimensionality, $xT = (x_1,\ldots,x_{j-1},\lambda x_j + (1-\lambda)x_k, x_{j+1},\ldots,x_{k-1},\lambda x_k + (1-\lambda)x_j, x_{k+1},\ldots,x_m)$.

T-transformations and majorization are closely linked by the following lemma (see [86]).

Lemma 3.4.1 *For $w, w' \in \mathbb{R}^m$, $w \succ w'$ if and only if w' can be derived from w by successive applications of at most $m-1$ T-transformations.*

One of the fundamental theorems in the theory of majorization is the following.

Theorem 3.4.2 *(Hardy, Littlewood and Pólya, 1929). For $w, w' \in \mathbb{R}^m, w \succ w'$ if and only if $w' = wP$, for some doubly stochastic matrix P.*

Schur-Convex A real-valued function ϕ defined on a set $\mathcal{A} \subset \mathcal{R}^n$ is said to be Schur-convex on $\mathcal{A}$ if

$$x \prec y \ \text{ on } \ \mathcal{A} \implies \phi(x) \leq \phi(y).$$

3.4.2 Weighted Single-choice Games

In this section we study the classical balls-into-bins game where every ball has only one random choice. Let w and w' be two $m-$dimensional weight vectors. Recall that $S_i(w)$ is defined to be the random variable counting the cumulative loads of the i largest bins after allocating w. In this section we show that, if there exists a majorization order between two weight vectors w and w', the same order holds for $E[S_i(w)]$ and $E[S_i(w')]$. This implies that, if w majorizes w', the expected maximum load after allocating w is larger than or equal to the expected maximum load after allocating w'.

Note that in the single-choice game, the final load distribution does not depend upon the order in which the balls are allocated. From Lemma 3.4.1 we know that, if $w \succ w'$, then w' can be derived from w by applying at most $m-1$ T-transformations. Thus, it is sufficient to show the case in which w' can be derived from w by applying one T-transformation, which is what we do in Lemma 3.4.4. First, we give a simple lemma which is used later in Lemma 3.4.4.

Lemma 3.4.3 *Consider two vectors $u = (u_1, u_2, \ldots, u_n)$ and $v = (v_1, v_2, \ldots, v_n)$, and assume u, v are sorted in non-increasing order, i.e., $u_1 \geq u_2 \geq \ldots \geq u_n$ and $v_1 \geq v_2 \geq \ldots \geq v_n$. For any $t \geq 0$, define $u \diamond \{t\}$ to be the (sorted) vector obtained from u by appending a new dimension with coordinate value t to u. Similarly we define $v \diamond \{t\}$. If $u \succ v$, then $u \diamond \{t\} \succ v \diamond \{t\}$.*

Proof. Let $u' = \{u'_1, u'_2, \ldots, u'_{n+1}\} = u \diamond \{t\}$, $v' = \{v'_1, v'_2, \ldots, v'_{n+1}\} = v \diamond \{t\}$. Assume u' and v' are sorted in non-increasing order, i.e., $u'_1 \geq u'_2 \geq \ldots \geq u'_{n+1}$, $v'_1 \geq v'_2 \geq \ldots \geq v'_{n+1}$. According to the definition of majorization, for any $i \in [1, n]$, we have $\sum_{j=1}^{i} u_j \geq \sum_{j=1}^{i} v_j$. Now fix $i \in [1, n+1]$, we have to show that $\sum_{j=1}^{i} u'_j \geq \sum_{j=1}^{i} v'_j$. Depending on the size of t, there are four cases:

1. $t \leq \min\{u_i, v_i\}$. Since t is neither one of the i largest items in u' nor v', the majorization at position i is still preserved.
2. $t \geq \max\{u_i, v_i\}$. In this case, t is one of the i largest items in both u' and v', which again means that the majorization at position i is still preserved.
3. $u_i \geq t \geq v_i$. Now t is one of the i largest items in v', but not in u'. We get

$$\sum_{j=1}^{i} u'_j = \sum_{j=1}^{i} u_j = \sum_{j=1}^{i-1} u_j + u_i \geq \sum_{j=1}^{i-1} v_j + t = \sum_{j=1}^{i} v'_j.$$

4. $v_i \geq t \geq u_i$. Similarly, t is one of the i largest items in u', but not in v'.

$$\sum_{j=1}^{i} u'_j = t + \sum_{j=1}^{i-1} u_j \geq u_i + \sum_{j=1}^{i-1} u_j = \sum_{j=1}^{i} u_j \geq \sum_{j=1}^{i} v_j = \sum_{j=1}^{i} v'_j.$$

By the definition of majorization, we conclude that $u' \succ v'$, i.e., $u \diamond \{t\} \succ v \diamond \{t\}$. □

Now we examine the case where the weight vectors w and w' differ by a single T-transformation.

Lemma 3.4.4 *If* $w \overset{T}{\Longrightarrow} w'$ *(i.e.* $w \succ w'$*), then* $E[S_i(w)] \geq E[S_i(w')]$ *for all* $i \in [n]$.

Proof. Let $w = (w_1, \ldots, w_m)$. According to the definition of a T-transformation, for some $0 \leq \lambda \leq 1$, we have

$$w' = wT = (w_1, \ldots, w_{j-1}, \lambda w_j + (1-\lambda)w_k, w_{j+1}, \ldots, w_{k-1}, \lambda w_k + (1-\lambda)w_j, w_{k+1}, \ldots, w_m).$$

We define $y_j = \max\{\lambda w_j + (1-\lambda)w_k, \lambda w_k + (1-\lambda)w_j\}, y_k = \min\{\lambda w_j + (1-\lambda)w_k, \lambda w_k + (1-\lambda)w_j\}$. Note that $w_j + w_k = y_j + y_k$, and $w_j \geq y_j \geq y_k \geq w_k$. Let $\Delta = w_j - y_j = y_k - w_k$.

Since the final allocation does not depend on the order in which the balls are allocated, we can assume in the following that both w_j, w_k and y_j, y_k are allocated in the last two steps. Now fix the random choices for the first $m-2$ balls and let $\ell = (\ell_1, \ldots, \ell_n)$ be the resulting normalized load vector. Let $\Omega^2 = [n]^2$ be the set of random choices of the last two balls. Note that every random choice in Ω^2 occurs with same probability $1/n^2$.

Now fix a pair of choices (p, q) for the last two balls and define $L(\ell, (w_j, p), (w_k, q))$ as the load vector after placing the ball with weight w_j into the bin with rank p in ℓ, and the ball with weight w_k into the bin with rank q in ℓ. (Note, after the allocation of w_j the order of the bins might change but q still refers to the old order. Let $S_i(\ell, (w_j, p), (w_k, q))$ be the cumulative load of the i largest bins of $L(\ell, (w_j, p), (w_k, q))$. Similarly we define $L(\ell, (y_j, p), (y_k, q))$ and $S_i(\ell, (y_j, p), (y_k, q))$. In the following we compare the two choices (p, q) and (q, p) with each other and show that for all ℓ

$$S_i(\ell, (w_j, p), (w_k, q)) + S_i(\ell, (w_j, q), (w_k, p)) \geq S_i(\ell, (y_j, p), (y_k, q)) + S_i(\ell, (y_j, q), (y_k, p)).$$

Since we compute expected values over all pairs (p, q), this shows that the *expected* cumulative loads of the i largest bins of both allocations also obey the same order (see Definition 3.4.1).

For $p = q$, the two balls are allocated to the same bin, so the resulting load vectors are identical. Therefore $S_i(\ell, (w_j, p), (w_k, q)) = S_i(\ell, (y_j, p), (y_k, q))$. The next lemma considers the case $p < q$. □

Lemma 3.4.5 *For any normalized ℓ and $\forall i, p, q \in [n]$ with $p < q$, we have*

$$S_i(\ell, (w_j, p), (w_k, q)) + S_i(\ell, (w_j, q), (w_k, p)) \geq S_i(\ell, (y_j, p), (y_k, q)) + S_i(\ell, (y_j, q), (y_k, p)).$$

Proof. In the following P and Q are the bins with rank p and q, respectively (after the allocation of the first $m-2$ balls). The load of all other bins remains the same. The following observation (given without proof) compares the load of Q and P for the different possible allocations of w_k, w_j, y_k and y_j with each other. Recall that because ℓ is normalized (and $p < q$) we know that $\ell_p \geq \ell_q$.

Observation 3.4.6 *For $w_j \geq y_j \geq y_k \geq w_k$ and $\ell_p \geq \ell_q$*

$$\begin{aligned} w_j + \ell_p &\geq \max\{w_k + \ell_p,\ w_k + \ell_q,\ w_j + \ell_q\} \\ y_j + \ell_p &\geq \max\{y_k + \ell_p,\ y_k + \ell_q,\ y_j + \ell_q\} \\ w_k + \ell_q &\leq \min\{w_j + \ell_q,\ w_j + \ell_p,\ w_k + \ell_p\} \\ y_k + \ell_q &\leq \min\{y_j + \ell_q,\ y_j + \ell_p,\ y_k + \ell_p\}. \end{aligned}$$

Unfortunately, we can not rank $w_j + \ell_q$ and $w_k + \ell_p$, and $y_j + \ell_q$ and $y_k + \ell_p$ so far. To do that we consider in the following the two 2-dimensional vectors $(w_j + \ell_q, w_k + \ell_p)$ and $(y_k + \ell_p, y_j + \ell_q)$. Since

$$(w_j + \ell_q) + (w_k + \ell_p) = (y_k + \ell_p) + (y_j + \ell_q) \tag{3.1}$$

one of them majorizes the other. This gives the following two cases.

Case I: $(w_j + \ell_q, w_k + \ell_p) \succ (y_j + \ell_q, y_k + \ell_p)$
In this case we can iteratively apply Lemma 3.4.3 to show that $L(\ell, (w_j, q), (w_k, p)) \succ L(\ell, (y_j, q), (y_k, p))$. Hence, for any $i \in [n]$ we have

$$S_i(\ell, (w_j, q), (w_k, p)) \geq S_i(\ell, (y_j, q), (y_k, p)).$$

Since $w_j \geq y_j$ and $y_k \geq w_k$ we have $w_j + \ell_p \geq y_j + \ell_p$ and $y_k + \ell_q \geq w_k + \ell_q$. From Observation 3.4.6(4) we know $y_j + \ell_p \geq y_k + \ell_q$. If we now allocate w_j and y_j to ℓ_q, and w_k and y_k to ℓ_p, we still have $(w_j + \ell_p, w_k + \ell_q) \succ (y_j + \ell_p, y_k + \ell_q)$. This again yields

$$S_i(\ell, (w_j, p), (w_k, q)) \geq S_i(\ell, (y_j, p), (y_k, q)).$$

Since for both pairs (p,q) and (q,p) the resulting load vector with y_j and y_k is majorized by its counterpart with the balls w_j and w_k, the proof of this case is finished.

A simple example for this case is the following: $\ell_p = 3$, $\ell_q = 2$, $w_j = 6$, $w_k = 1$, $y_j = 4$, and $y_k = 3$. For this case we have, $(w_j + \ell_q, w_k + \ell_p) = (8,4) \succ (7,5) = (y_k + \ell_p, y_j + \ell_q)$ and $(w_j + \ell_p, w_k + \ell_q) = (9,3) \succ (6,6) = (y_k + \ell_p, y_j + \ell_q)$.

Case II: $(w_j + \ell_q, w_k + \ell_p) \prec (y_j + \ell_q, y_k + \ell_p)$
In this case we have $L(\ell,(w_j,q),(w_k,p)) \prec L(\ell,(y_j,q),(y_k,p))$. Hence, we have to consider the two pairs of choices (p,q) and (q,p) together to show our result. We get

$$S_i(\ell,(w_j,p),(w_k,q)) + S_i(\ell,(w_j,q),(w_k,p) \geq S_i(\ell,(y_j,p),(y_k,q)) + S_i(\ell,(y_j,q),(y_k,p)).$$

Since $\max\{w_j + \ell_q, w_k + \ell_p\} \leq \max\{y_k + \ell_p, y_j + \ell_q\}$ and $w_j + \ell_q \geq y_j + \ell_q$, we have $y_k + \ell_p \geq y_j + \ell_q$. This results in $y_k + \ell_p \geq w_j + \ell_q$. Using Equation (3.1) we get $y_j + \ell_q \leq w_k + \ell_p$. Hence, we can order the pairs:

$$y_k + \ell_p \geq w_j + \ell_q \text{ and } w_k + \ell_p \geq y_j + \ell_q. \tag{3.2}$$

Now, what happens if we consider the pair (q,p) which allocates w_j and y_j to Q, and w_k and y_k to P? Since $\ell_p \geq \ell_q$ and $w_j \geq y_j \geq y_k \geq w_k$ we have

$$w_j + \ell_p \geq y_j + \ell_p \geq y_k + \ell_q \geq w_k + \ell_q. \tag{3.3}$$

Now we consider three subcases depending on the order of p, q, and i.

Case II(A): Bin $y_k + \ell_p$ is not among the i largest bins in $L(\ell,(y_j,q),(y_k,p))$. From Equation (3.2) we know that $y_k + \ell_p$ is not smaller than $w_j + \ell_q$, $w_k + \ell_p$, or $y_j + \ell_q$. Hence, $y_j + \ell_q$ is also not among the i largest bins. Since the load of all bins except P and Q remain unchanged, $w_j + \ell_q$ and $w_k + \ell_p$ are also not among the i largest bins of $L(\ell,(w_j,q),(w_k,p))$. Thus, we have

$$S_i(\ell,(y_j,q),(y_k,p)) = S_i(\ell,(w_j,q),(w_k,p)).$$

Due to Equation (3.3) we have $(w_j + \ell_p, w_k + \ell_q) \succ (y_j + \ell_p, y_k + \ell_q)$. Similar to Case I we can show

$$S_i(\ell,(y_j,p),(y_k,q)) \leq S_i(\ell,(w_j,p),(w_k,q)).$$

Case II(B): Bin $y_j + \ell_q$ is one of the i largest bins in $L(\ell, (y_j, q), (y_k, p))$. From Equation (3.2) we know that $y_j + \ell_q$ is not larger than $w_j + \ell_q$, $w_k + \ell_p$, or $y_k + \ell_p$. Hence, $y_k + \ell_p$ is also among the i largest bins. Since the load of all bins except P and Q remain unchanged, $w_j + \ell_q$ and $w_k + \ell_p$ are also among the i largest bins of $L(\ell, (W_j, q), (w_k, p))$. Again, we have

$$S_i(\ell, (y_j, q), (y_k, p)) = S_i(\ell, (w_j, q), (w_k, p))$$

and

$$S_i(\ell, (y_j, p), (y_k, q)) \leq S_i(\ell, (w_j, p), (w_k, q)).$$

Case II(C): Bin $y_k + \ell_p$ is one of the i largest bins of $L(\ell, (y_j, q), (y_k, p))$, and bin $y_j + \ell_q$ is not. Due to Observation 3.4.6(2) and $w_j + \ell_p \geq y_j + \ell_p$, bin P must also be among the i largest bins of both $L(\ell, (w_j, p), (w_k, q))$ and $L(\ell, (y_j, p), (y_k, q))$, respectively. Similarly, using Observation 3.4.6(4) and $w_k + \ell_q \leq y_k + \ell_q$, we know that bin Q can not be among the i largest bins of $L(\ell, (w_j, p), (w_k, q))$ and $L(\ell, (y_j, p), (y_k, q))$. This gives us

$$S_i(\ell, (w_j, p), (w_k, q)) - S_i(\ell, (y_j, p), (y_k, q)) = (w_j + \ell_p) - (y_j + \ell_p) = \Delta. \tag{3.4}$$

Now it remains to compare $S_i(\ell, (w_j, q), (w_k, p))$ and $S_i(\ell, (y_j, q), (y_k, p))$ with each other. Since $y_j + l_q$ is not among the i largest bins in $S_i(\ell, (y_j, q), (y_k, p))$, we have $S_i(\ell, (y_j, q), (y_k, p)) = S_i(\ell, (y_k, p))$. In Observation 3.4.7 below we show that $S_i(\ell, (y_k, p)) - S_i(\ell, (w_k, p)) \leq \Delta$. Since $S_i(\ell, (w_j, q), (w_k, p)) \geq S_i(\ell, (w_k, p))$ we get

$$S_i(\ell, (y_j, q), (y_k, p)) - S_i(\ell, (w_j, q), (w_k, p)) \leq \Delta. \tag{3.5}$$

Equations (3.4) and (3.5) together give

$$S_i(\ell, (w_j, p), (w_k, q)) - S_i(\ell, (y_j, p), (y_k, q)) \geq S_i(\ell, (y_j, q), (y_k, p)) - S_i(\ell, (w_j, q), (w_k, p)).$$

□

Observation 3.4.7 $S_i(\ell, (y_k, p)) - S_i(\ell, (w_k, p)) \leq \Delta$.

Proof. The proof is split into three cases. If bin P is among the i largest bins in $L(\ell, (w_k, p))$, it is also among the i largest bins in $L(\ell, (y_k, p)$ $(y_k > w_k)$. In this case

$$S_i(\ell, (y_k, p)) - S_i(\ell, (w_k, p)) = \Delta.$$

If bin P is not among the i largest bins in $L(\ell,(y_k,p))$ it is also not among the the i largest bins in $L(\ell,(w_k,p))$. Hence,

$$S_i(\ell,(y_k,p)) - S_i(\ell,(w_k,p)) = 0.$$

In the last case, bin P is among the i largest bins in $L(\ell,(y_k,p))$, but not in $L(\ell,(w_k,p))$. In this case we have

$$S_i(\ell,(y_k,p)) - S_i(\ell,(w_k,p)) \leq (y_k + \ell_p) - (w_k + \ell_p) = \Delta.$$

(Or, it follows directly since bin P is the only bin who gets different load in $L(\ell,(y_k,p))$ and $L(\ell,(w_k,p))$, hence the difference is at most Δ.) □

The iterative application of Lemma 3.4.4 can now be used to generalize the majorization result for vectors that only differ by a single T-transformation to vectors that differ by several T-transformations. This results in the following theorem:

Theorem 3.4.8 *If $w \succ w'$, then $E[S_i(w)] \geq E[S_i(w')]$ for all $i \in [n]$.*

Proof. By Lemma 3.4.1, if $w \succ w'$, then w' can be derived from w by applying at most $m-1$ T-transformations. In other words, letting $k \in \{1,\ldots,m-1\}$ be the total number of T-transformations, there must exist $k-1$ m-dimensional vectors $v_1,\ldots,v_{k-1}$, such that

$$w \overset{T}{\Longrightarrow} v_1 \overset{T}{\Longrightarrow} \ldots \overset{T}{\Longrightarrow} v_{k-1} \overset{T}{\Longrightarrow} w'.$$

Similar to $S_i(w)$ we define $S_i(v_j)$, $j \in \{1,\ldots,k-1\}$. Iteratively applying Lemma 3.4.1, we get

$$E[S_i(w)] \geq E[S_i(v_1)] \geq \ldots \geq E[S_i(v_{k-1})] \geq E[S_i(w')].$$

□

Finally, it is clear that the uniform weight vector is majorized by all other vectors with same dimension and same total weight. Using Theorem 3.4.8, we get the following corollary.

Corollary 3.4.9 *Let $w = (w_1,\ldots,w_m)$, $W = \sum_{i=1}^m w_i$, and $w' = (\frac{W}{m},\ldots,\frac{W}{m})$. For all $i \in [n]$, we have $E[S_i(w)] \geq E[S_i(w')]$.*

Proof. Note that $w' = wP$, where $P = (p_{ij})$ and $p_{ij} = 1/m$ $\forall i,j \in [m]$. Clearly P is a doubly stochastic matrix. Hence by Lemma 3.4.2, $w \succ w'$. Consequently, from Theorem 3.4.8 we have $E[S_i(w)] \geq E[S_i(w')]$. □

Theorem 3.4.8 also shows that an allocation of a large number of small balls with total weight W ends up with a smaller expected load than the allocation of a smaller number of balls with the same total weight. Note that in the next corollary the relation $w \succ w'$ must be treated somewhat loosely because the vectors do not necessarily have the same length, but the meaning should be clear, namely that $\sum_{i=1}^{j} w_i \geq \sum_{i=1}^{j} w'_i$ for all $j \in [m]$.

Corollary 3.4.10 *Let $w = (w_1, \ldots, w_m)$ and $W = \sum_{i=1}^{m} w_i$. Suppose that $w' = (w'_1, \ldots, w'_{m'})$ with $m \leq m'$, and also that $W = \sum_{i=1}^{m} w'_i$. If $w \succ w'$ we have $E[S_i(w)] \geq E[S_i(w')]$ for all $i \in [n]$.*

Proof. Simply add zeros to w until it has the same dimension than w'. □

It is easy to see that we can generalize the result to other probability distributions that are used to choose the bins.

Corollary 3.4.11 *If $w \succ w'$, and the probability that a ball is allocated to bin b_i, $1 \leq i \leq n$, is the same for all balls, then we have $E[S_i(w)] \geq E[S_i(w')]$ for all $i \in [n]$.*

Remark The work was submitted to a journal once. One of the anonymous referees gave some very nice idea to simplify the proof of Theorem 3.4.8. See Appendix 6.2 for an alternative proof. The referee also asked whether Lemma 3.4.4 could be generalized to the following. If $w \overset{T}{\Longrightarrow} w'$, does $S_i(w)$ stochastically dominate [3] $S_i(w')$? If this argument was true, it would directly imply Lemma 3.4.4. Unfortunately it is not the case. Consider two weight vectors $w = (5, 4, 2)$ and $w' = (5, 3, 3)$ with the same total weight 11. After allocating them into n bins, we get,

$$\Pr[S_1(w) \geq 8] = 1/n < 2/n - 1/n^2 = \Pr[S_1(w') \geq 8].$$

Thus, $S_1(w)$ does *not* stochastically dominate $S_1(w')$. This implies that Lemma 3.4.4 is probably the best result one can expect.

3.4.3 Weighted Multiple-choice Games

In the first sub-section we show that for multiple-choice games it is not always better to allocate uniform balls. For $m \gg n$ we construct a set of weighted balls that ends up with a

[3] For random variables X and Y, we say X stochastically dominates Y, written $X \succ Y$, or $Y \prec X$, if $\forall k \in \mathbb{R}$, $\Pr[X \geq k] \geq \Pr[Y \geq k]$.

smaller maximum load than a set of uniform balls with the same total weight. The second sub-section considers the case where m is not much larger than n. As we will argue in the beginning of that section, it appears that it may not be possible to use the majorization technique to get tight results for the weighted multiple-choice game. This is due to the fact that the order in which weighted balls are allocated is crucial, but the majorization order is not necessarily preserved for weighted balls in the multiple-choice game (in contrast to [13] for uniform balls). We discuss several open questions and give some weight vectors that result in a smaller expected maximum load than uniform vectors with the same total weight.

Large Number of Balls

We compare two allocations, $\mathcal{A}$ and $\mathcal{B}$. In $\mathcal{A}$ we allocate $m/2$ balls of weight 3 each and thereafter $m/2$ balls of weight 1 each, using the multiple-choice strategy. Allocation $\mathcal{B}$ is the uniform counterpart of $\mathcal{A}$ where all balls have weight 2. We show that the expected maximum load in $\mathcal{A}$ is strictly smaller than that in $\mathcal{B}$. We will use the *short term memory property* stated below in Lemma 3.4.12. See [20] for a proof. Basically, this property says that after allocating a sufficiently large number of balls, the load depends on the last poly(n) many balls only. If m is now chosen large enough (but polynomially large in n suffices), then the maximum load is (w.h.p. upper bounded by $2m/n + \log\log n$. In the case of balls with weight 2, the maximum load is w.h.p.upper bounded by $2m/n + 2\log\log n$. Since [20] gives only upper bounds on the load, we can not use the result directly. We introduce two auxiliary allocations named $\mathcal{C}$ and $\mathcal{D}$. Allocation $\mathcal{C}$ is derived from Allocation $\mathcal{A}$, and $\mathcal{D}$ is derived from $\mathcal{B}$. The only difference is that in allocations $\mathcal{C}$ and $\mathcal{D}$ we allocate the first $m/2$ balls optimally (i.e. we always place the balls into the least loaded bins). In Lemma 3.4.16 we first show that the maximum loads of $\mathcal{A}$ and $\mathcal{C}$ will be nearly indistinguishable after allocating all the balls. Similarly, the maximum loads of $\mathcal{B}$ and $\mathcal{D}$ will be nearly indistinguishable. Moreover, we show that the expected maximum load in $\mathcal{D}$ is larger than that in $\mathcal{C}$. Then we can show that the expected maximum load in $\mathcal{A}$ is smaller than that in $\mathcal{B}$ (Theorem 3.4.17). For an overview of the four systems, we refer to Table 3.1.

To state the short memory property we need one more definition. For any two random variables X and Y defined jointly on the same sample space, the *variation distance* between

Table 3.1: Allocations $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$

	First $m/2$ balls		Last $m/2$ balls	
Allocations	ball weights	algorithm	ball weights	algorithm
$\mathcal{A}$	3	Greedy[d]	1	Greedy[d]
$\mathcal{B}$	2	Greedy[d]	2	Greedy[d]
$\mathcal{C}$	3	Optimal	1	Greedy[d]
$\mathcal{D}$	2	Optimal	2	Greedy[d]

$\mathcal{L}(X)$ (the "law", or distribution, of X) and $\mathcal{L}(Y)$ is defined as

$$\|\mathcal{L}(X) - \mathcal{L}(Y)\| = \sup_A |\Pr(X \in A) - \Pr(Y \in A)|.$$

The following lemma is from [20, Corollary 1].

Lemma 3.4.12 *Suppose $L_0 = (\ell_1, \ldots, \ell_n)$ is an arbitrary normalized load vector describing an allocation of m balls into n bins. Define $\Delta = \ell_1 - \ell_n$ to be the maximum load difference in L_0. Let L'_0 be the load vector describing the optimal allocation of the same number of balls to n bins. Let L_k and L'_k, respectively, denote the vectors obtained after inserting k further balls to both allocations using the multiple-choice algorithm. Then for $k \geq n^5 \cdot \Delta$*

$$|| \mathcal{L}(L_k) - \mathcal{L}(L'_k) || \leq k^{-\alpha}$$

where α is an arbitrary constant.

Intuitively, Lemma 3.4.12 indicates that given any configuration with maximum difference Δ, in $\Delta \cdot \text{poly}(n)$ steps the allocation "forgets" the difference, i.e., the allocation is nearly indistinguishable from the allocation obtained by starting from a completely balanced allocation. This is in contrast to the single-choice game requiring $\Delta^2 \cdot \text{poly}(n)$ steps in order to "forget" a load difference Δ (see [20]).

Lemma 3.4.13 *Suppose we allocate m balls to n bins using Greedy[d] with $d \geq 2$, $m \gg n$. Then the number of bins with load at least $m/n + i + \gamma$ is bounded above by $n \cdot \exp(-d^i)$, w.h.p, where γ denotes a suitable constant. In particular, the maximum load is w.h.p.*

$$\frac{m}{n} + \frac{\log \log n}{\log d} \pm \Theta(1).$$

Proof. The result that the maximum load is at most $m/n + \log\log n/\log d + \Theta(1)$ has been shown in [20]. To show the lower bound we first recall two results shown in [13]. First, let u and v be two positive integer vectors such that $u_1 \geq u_2 \geq \ldots \geq u_n$ and $v_1 \geq v_2 \geq \ldots \geq v_n$. Azar et al. [13] show that if $u \succ v$, then also $u + e_i \succ v + e_i$, where e_i is the ith unit vector. Now let u, v be two vectors with same total weight. Denote by u' and v' the load vectors obtained by allocating a unit-size ball b into two allocations having initial loads u, v respectively. Then Azar et al. [13] show the following theorem:

Theorem 3.4.14 *If $u \succ v$, there is a coupling of two allocations with respect to the allocation of b such that $u' \succ v'$.*

We consider two allocations $\mathcal{E}$ and $\mathcal{F}$. In $\mathcal{E}$ we allocate m balls into n bins using Greedy[d], while in $\mathcal{F}$, we first place $m - n$ balls optimally, and then allocate the remaining n balls by Greedy[d]. Clearly after allocating the first $m - n$ balls, the normalized load vector of $\mathcal{E}$ always majorizes the normalized load vector of $\mathcal{F}$. Applying Theorem 3.4.14 on the last n balls, we see that $\mathcal{E} \succ \mathcal{F}$ in a stochastic sense. Since the maximum load in $\mathcal{F}$ is known to be lower bounded by $m/n + \log\log n/\log d - \Theta(1)$ w.h.p.[13]), the same lower bound holds for the maximum load of $\mathcal{E}$. □

Let $L_i(\mathcal{A})$ (or $L_i(\mathcal{B})$, $L_i(\mathcal{C})$, $L_i(\mathcal{D})$) be the maximum load in Allocation $\mathcal{A}$ (respectively, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$) after the allocation of the first i balls. If we refer to the maximum load after the allocation of all m balls we will simply write $L(\mathcal{A})$ (or $L(\mathcal{B})$, $L(\mathcal{C})$, $L(\mathcal{D})$). Lemma 3.4.16 below compares the load of the four allocations described in Table 3.1. First, we give a lemma stating that, given two random variables, a small variation distance implies a small difference between their expectations.

Lemma 3.4.15 *Let X and Y be two discrete random variables sharing the same sample space. Let ζ be the maximum possible value of X and Y. Then,*

$$|E[X] - E[Y]| \leq \zeta \cdot ||\mathcal{L}(X) - \mathcal{L}(Y)||.$$

Proof. Let $G = \{k | \Pr(X = k) > \Pr(Y = k)\}$, $S = \{k | \Pr(X = k) < \Pr(Y = k)\}$. Due to choice of G and S,

$$\sum_{k \in G} (\Pr(X = k) - \Pr(Y = k)) = \sum_{k \in S} (\Pr(Y = k) - \Pr(X = k)).$$

Hence,

$$
\begin{aligned}
&|E[X] - E[Y]| \\
&= \left| \sum_k \left(\left(\Pr(X = k) - \Pr(Y = k) \right) \cdot k \right) \right| \\
&= \left| \sum_{k \in G} \left(\left(\Pr(X = k) - \Pr(Y = k) \right) \cdot k \right) - \sum_{k \in S} \left(\left(\Pr(Y = k) - \Pr(X = k) \right) \cdot k \right) \right| \\
&\leq \max \left\{ \sum_{k \in G} \left(\left(\Pr(X = k) - \Pr(Y = k) \right) \cdot k \right), \sum_{k \in S} \left(\left(\Pr(Y = k) - \Pr(X = k) \right) \cdot k \right) \right\} \\
&\leq \max \left\{ \zeta \cdot \sum_{k \in G} \left(\Pr(X = k) - \Pr(Y = k) \right), \zeta \cdot \sum_{k \in S} \left(\Pr(Y = k) - \Pr(X = k) \right) \right\} \\
&= \zeta \cdot \sum_{k \in G} \left(\Pr(X = k) - \Pr(Y = k) \right) \\
&\leq \zeta \cdot \sup_A \left| \Pr(X \in A) - \Pr(Y \in A) \right| = \zeta \cdot \| \mathcal{L}(X) - \mathcal{L}(Y) \|.
\end{aligned}
$$

□

Lemma 3.4.16 *Let $m = \Omega(n^7)$.*

(a) $|E[L(\mathcal{A})] - E[L(\mathcal{C})]| \leq m^{-\beta}$, *where β is an arbitrary constant.*

(b) $|E[L(\mathcal{B})] - E[L(\mathcal{D})]| \leq m^{-\beta'}$, *where β' is an arbitrary constant.*

(c) $E[L(\mathcal{D})] - E[L(\mathcal{C})] \geq \frac{\log \log n}{\log d} - \Theta(1)$.

Proof. Part (a). By Lemma 3.4.13 we get w.h.p.

$$L_{\frac{m}{2}}(\mathcal{A}) \leq 3 \cdot \left(\frac{m}{2n} + \left(\frac{\log \log n}{\log d} \right) \right) + \Theta(1).$$

Using the pigeonhole principle, the maximum load difference Δ is at most $3 \cdot n \cdot \log \log n / \log d +$ $\Theta(n)$ — assume the worst case where $n - 1$ of the n bins are maximally loaded, and only one bin is below average.

Since $m/2 = \Omega(n^7) > n^5 \Delta$, by Lemma 3.4.12, the Greedy[d] algorithm has "short memory". In other words, after allocating the remaining $m/2$ balls of weight one, $\mathcal{A}$ and $\mathcal{C}$ will become almost indistinguishable. Moreover, we note that the variation distance of two random vectors is certainly no bigger than that of their respective maxima, thus

$$\| \mathcal{L}(L(\mathcal{A})) - \mathcal{L}(L(\mathcal{C})) \| \leq \left(\frac{m}{2} \right)^{-\alpha}$$

where α is an arbitrary constant. It is clear that the maximal possible loads of both allocations $\mathcal{A}$ and $\mathcal{C}$ are $2m$ (if we allocate all the balls into one bin). By Lemma 3.4.15,

$$|E[L(\mathcal{A})] - E[L(\mathcal{C})]| \leq \left(\frac{m}{2}\right)^{-\alpha} \cdot 2m \leq m^{-\beta}$$

as long as we choose $\alpha = \frac{(1+\beta)\log_2 m+1}{\log_2 m-1}$.

Part (b). This can be shown similar to part (a).

Part (c). The deviation of the maximum load from the average in Allocation $\mathcal{D}$ is exactly twice that of $\mathcal{C}$, or

$$E[L(\mathcal{D})] - \frac{2m}{n} = 2 \cdot \left(E[L(\mathcal{C})] - \frac{2m}{n}\right).$$

Hence,

$$E[L(\mathcal{D})] - E[L(\mathcal{C})] = E[L(\mathcal{C})] - \frac{2m}{n}.$$

By Lemma 3.4.13, the maximum load of Allocation $\mathcal{C}$ is at least $\frac{2m}{n} + \frac{\log\log n}{\log d} - \Theta(1)$ w.h.p.Hence,

$$E[L(\mathcal{D})] - E[L(\mathcal{C})] = E[L(\mathcal{C})] - \frac{2m}{n} \geq \frac{\log\log n}{\log d} - \Theta(1).$$

□

Finally, we present the main result of this section, showing that uniform balls do not necessarily minimize the maximum load in the multiple-choice game.

Theorem 3.4.17 $E[L(\mathcal{B})] \geq E[L(\mathcal{A})] + \frac{\log\log n}{\log d} - \Theta(1)$.

Proof. Clearly

$$E[L(\mathcal{B})] - E[L(\mathcal{A})] = (E[L(\mathcal{D})] - E[L(\mathcal{C})]) - (E[L(\mathcal{A})] - E[L(\mathcal{C})]) + (E[L(\mathcal{B})] - E[L(\mathcal{D})]).$$

Since the difference between $(E[L(\mathcal{A})] - E[L(\mathcal{C})])$ and $(E[L(\mathcal{B})] - E[L(\mathcal{D})])$ is at most $m^{-\beta}$ (Lemma 3.4.16), we conclude that

$$E[L(\mathcal{B})] - E[L(\mathcal{A})] \geq \frac{\log\log n}{\log d} - \Theta(1) - m^{-\beta} - m^{-\beta'} \geq \frac{\log\log n}{\log d} - \Theta(1).$$

□

Majorization Order for Arbitrary Number of Balls

In this section we consider the Greedy[2] process applied on weighted balls, but most of the results can be generalized to the Greedy[d] process for $d > 2$. Just to remind you, in the Greedy[2] process each ball sequentially picks independently uniformly at random two bins and the current ball is allocated in the least loaded of the two bins (ties can be broken arbitrarily). This means, of course, that a bin with relative low load is more likely to get an additional ball than one of the highly loaded bins.

Another way to model the Greedy[d] process is the following: Assume that the load vector of the bins are normalized, i.e. $\ell_1 \geq \ell_2 \geq \cdots \geq \ell_n$. If we now place an additional ball into the bins, the ball will be allocated to bin i with probability $(i^d - (i-1)^d)/n^d$, since all d choices have to be among the first i bins, and at least one choice has to be i. For $d = 2$ this simplifies to $(2i-1)/n^2$. Hence, in this fashion, the process can be viewed as a "one choice process", provided the load vector is re-normalized after the allocation of each ball. This means that the load distribution of the bins highly depends on the order in which the balls are allocated.

Unfortunately, the dependence of the final load distribution on the order in which the balls are allocated makes it very hard to get tight bounds using the majorization technique together with T-transformations. Theorem 3.4.8 highly depends on the fact that we can assume that w_j and w_k (y_j and y_k) are allocated at the very end of the process, an assumption that can not be used in the multiple-choice game. In order to use T-transformations for multiple-choice games, we would again need a result that shows that the majorization order is preserved when we add more (similar) balls into the allocation. We need a result showing that if $\mathcal{A} \succ \mathcal{B}$ and we add an additional ball to both $\mathcal{A}$ and $\mathcal{B}$, after the allocation we still have $\mathcal{A}' \succ \mathcal{B}'$ (where $\mathcal{A}'$ and $\mathcal{B}'$ denote the new allocations with the one additional ball). While this is true for uniform balls (see [13]), this is not necessarily true for weighted balls and the multiple-choice game. In the following sections we study the majorization order for weighted multiple choice games, and the effect that the the allocation order or the number of balls have on the final load distribution.

Majorization Order The following easy example shows that the majorization order need not be preserved for weighted balls in the multiple-choice case. Let $\mathcal{A} = (7, 6, 5)$ and $\mathcal{B} = (7, 5.8, 5.2)$. If we now allocate one more ball with weight $w = 2$ into both systems (using the Greedy[2] algorithm), with probability 5/9 the ball is allocated to the third bin in both

allocations and we have $\mathcal{A}' = (7,7,6)$ and $\mathcal{B}' = (7.2,7,5.8)$, hence $\mathcal{B}' \succ \mathcal{A}'$. Alternatively, with probability $1/3$ the ball is allocated to the second bin in each allocation resulting in load vectors $\mathcal{A}' = (8,7,5)$ and $\mathcal{B}' = (7.8,7,5.2)$. Finally, with probability $1/n^2$ the ball is allocated to the first bin resulting in load vectors $\mathcal{A}' = (9,6,5)$ and $\mathcal{B}' = (9,5.8,5.2)$. In both cases we still have $\mathcal{A}' \succ \mathcal{B}'$. This shows that after the allocation of one additional ball using Greedy[2], the majorization relation can turn around. Note that the load distributions of $\mathcal{A}$ and $\mathcal{B}$ are not "atypical", but they can easily come up using Greedy[2].

The next lemma gives another example showing that the majorization relation need *not* be preserved for weighted balls in the multiple-choice game. The idea is that we can consider two allocations $\mathcal{C}$ and $\mathcal{D}$ where $\mathcal{C} \succ \mathcal{D}$, but by adding one additional ball (with large weight w), we then have $E[S_1(\mathcal{D}')] \geq E[S_1(\mathcal{C}')]$. It is easy to generalize the lemma to cases where w is not larger than the maximum bin load to show that the majorization relation need not be preserved.

Lemma 3.4.18 *Let v and u be two (normalized) load vectors with $v \overset{T}{\Longrightarrow} u$ (so $v \succ u$). u and v have same total weight and $u \neq v$. Let w be the weight of an additional ball with $w > v_1$. Let v', u' be the new (normalized) load vectors after allocating the additional ball into v and u. Then we have $E[S_1(u')] > E[S_1(v')]$.*

Proof. First we assume $v \overset{T}{\Longrightarrow} u$. Then, by the property of T-transformations, there must exist two bins with rank $j, k \in \mathbb{Z}^+$, $j < k$, such that $v_j > u_j > u_k > v_k$, and for $\forall i \neq j, k$, that $u_i = v_i$. Besides, we have $v_j - u_j = u_k - v_k > 0$. We observe that, since $w > v_1 \geq u_1$, the destination of the new ball immediately becomes the maximum loaded bin in both allocations. Since the probability to place the new ball on top of the i-th largest bin in both allocations is $\frac{i^d-(i-1)^d}{n^d}$ we get

$$\begin{aligned} E[S_1(u')] - E[S_1(v')] &= \sum_{i=1}^{n} \frac{i^d - (i-1)^d}{n^d} \cdot (u_i - v_i) \\ &= \frac{j^d - (j-1)^d}{n^d} \cdot (u_j - v_j) + \frac{k^d - (k-1)^d}{n^d} \cdot (u_k - v_k) \\ &> 0. \end{aligned}$$

Here the second equation holds since $\forall i \notin \{j, k\}$, $u_i = v_i$. The last inequality is due to the facts that $j < k$ and $v_j - u_j = u_k - v_k > 0$. □

Remark: We feel it necessary to point out that the preceding lemma applies only to the largest elements of u' and v'. It is possible that after the allocation of the new ball we

could have $E[S_2(u')] > E[S_2(v')]$ or the reverse inequality $E[S_2(v')] > E[S_2(u')]$. Recall that $E[S_2(u')]$ is the expected sum of the largest two elements.

For example, (using the Greedy[2] algorithm) if we take $v = (7,7,3), u = (7,5,5)$, and $w = 20$, then the first inequality holds. It is easy to check that $E[S_2(u')] = 32 > 31\frac{7}{9} = E[S_2(v')]$. On the other hand, using the vectors $v = (100,1,1), u = (35,34,33)$, and a new ball having weight $w = 101$ we find that $E[S_2(v')] = 202 > 169\frac{4}{9} = E[S_2(u')]$. However, Lemma 3.4.18 tells us that $E[S_1(u')] \geq E[(S_1(v')]$ holds in both cases.

Lemma 3.4.18 and the example preceding that lemma both showed that a more unbalanced weight vector can end up with a smaller expected maximum load after the allocation of some additional (and similar) balls. However, in those cases we assumed that the number of bins is very small, or that one of the balls is very big. Simulation results show that for most weight vectors w, w' with $w \succ w'$ the expected maximum load after the allocation of w' is smaller than the one after the allocation of w. Unfortunately, we have been unable to show formal results along these lines.

3.4.4 Order of Allocating Balls

Another interesting question concerns the order of allocating balls under the multiple-choice game. In the case that $m \geq n$ we conjecture that if all the balls are allocated in decreasing order, the expected maximum is the smallest among all possible permutations. This is more or less intuitive since if we always allocate bigger balls first, the chances would be low to place the remaining balls in those bins which are already occupied by the bigger balls. However, we still do not know how to prove this conjecture. We can answer the peer question: what about if we allocate balls in increasing order? The next observation shows that the increasing order does *not* always yield the worst outcome.

Observation 3.4.19 *Fix a set of weighted balls. The expected maximum load is not necessarily maximized by allocating balls in increasing order using the Greedy[2] algorithm.*

Proof. We compare two allocations $\mathcal{A}$ and $\mathcal{B}$ both with n bins. Let $w_{\mathcal{A}} = \{1,2,1,5\}$, and $w_{\mathcal{B}} = \{1,1,2,5\}$ be two weight vectors (sequences of ball weights). Notice that $w_{\mathcal{B}}$ is a monotonically increasing sequence while $w_{\mathcal{A}}$ is not. After allocating the first three balls, observe that the possible outcomes for $\mathcal{A}$ and $\mathcal{B}$ are $(2,1,1,0,\ldots 0)$, $(3,1,0\ldots,0)$, $(2,2,0\ldots,0)$ and $(4,0,\ldots 0)$. We can calculate the probabilities for $\mathcal{A}$ and $\mathcal{B}$ to end up in

outcome $(2,2,0\ldots,0)$ are $(1-1/n^2)\cdot 3/n^2$ and $(1-1/n^2)\cdot 1/n^2$, respectively. Moreover, notice both $\mathcal{A}$ and $\mathcal{B}$ have the same probability to end up in outcome $(2,1,1,0,\ldots 0)$ and $(4,0,\ldots,0)$. Consequently, $\mathcal{B}$ has more (in fact, $(1-1/n^2)\cdot 2/n^2$) probability to end up in outcome $(3,1,0,\ldots,0)$ than $\mathcal{A}$, while $\mathcal{A}$ is more likely to end up in outcome $(2,2,0,\ldots,0)$.

Hence, after allocating the first three balls, $\mathcal{B}$ certainly majorizes $\mathcal{A}$. Since the last ball (with weight 5) is bigger than the loads of all bins in both $\mathcal{A}$ and $\mathcal{B}$ after allocating the first three balls, by Lemma 3.4.18 the expected maximum load after allocating $w_{\mathcal{A}}$ is bigger than that after allocating $w_{\mathcal{B}}$. □

Observation 3.4.20 *If $n=2$, the expected maximum load is not necessarily minimized by allocating balls in decreasing order using the Greedy[2] algorithm.*

Proof. We compare two allocations $\mathcal{A}$ and $\mathcal{B}$ both with $m\geq 5$ balls and $n=2$ bins. Let $w_{\mathcal{A}}=\{9,6,\ldots,6,5,4,4\}$ and $w_{\mathcal{B}}=\{9,6,\ldots,6,4,5,4\}$ be the corresponding weight vectors. Note that $w_{\mathcal{A}}$ is monotonically increasing while $w_{\mathcal{B}}$ is not. Let $L_{\mathcal{A}}$ (or $L_{\mathcal{B}}$) be the maximum load after allocating $\mathcal{A}$ (or $\mathcal{B}$ respectively) using the Greedy[2] algorithm. In the following we show $E[L_{\mathcal{A}}]>E[L_{\mathcal{B}}]$.

For any $t>0$, let $X_{\mathcal{A}}(t)$ (or $X_{\mathcal{B}}(t)$) be a random variable indicating the load vector after allocating the first t balls in $\mathcal{A}$(or $\mathcal{B}$, respectively). Furthermore $X_{\mathcal{A}}(m-3)=X_{\mathcal{B}}(m-3)$. For any $\ell=(\ell_1,\ell_2)\in X_{\mathcal{A}}(m-3)$, let $d(\ell)=|\ell_1-\ell_2|$. Note that $d(\ell)\geq 6$ or $d(\ell)=3$. We consider the following two cases.

Case 1. $d(\ell)\geq 6$. Note that the weights of the $(m-2)^{th}$ and $(m-1)^{th}$ balls in both systems are smaller than 6. Due to symmetry, exchanging these two balls will not affect the expected maximum loads. Consequently,

$$E[L_{\mathcal{A}}|X_{\mathcal{A}}(m-3)=\ell]=E[L_{\mathcal{B}}|X_{\mathcal{B}}(m-3)=\ell]. \tag{3.6}$$

Case 2. $d(\ell)=3$. In this case we can write $\ell=(y+3,y)$ for some $y>0$. Next we show that $E[L_{\mathcal{A}}|X_{\mathcal{A}}(m-3)=\ell]>E[L_{\mathcal{B}}|X_{\mathcal{B}}(m-3)=\ell]$. Simply enumerating all cases (See Figure 3.1), we get,

Note that $L_{\mathcal{A}}=\max\{X_{\mathcal{A}}(m)\}=y+311/32$ and $L_{\mathcal{B}}=\max\{X_{\mathcal{B}}(m)\}=y+305/32$. Consequently,

$$E[L_{\mathcal{A}}|X_{\mathcal{A}}(m-3)=\ell]=E[L_{\mathcal{B}}|X_{\mathcal{B}}(m-3)]+3/16. \tag{3.7}$$

$$\Pr[X_{\mathcal{A}}(m)|X_{\mathcal{A}}(m-3)=\ell] = \begin{cases} (y+16,y) & \text{w. p. } 1/64 \\ (y+12,y+4) & \text{w. p. } 6/64 \\ (y+8,y+8) & \text{w. p. } 9/64 \\ (y+9,y+7) & \text{w. p. } 36/64 \\ (y+11,y+5) & \text{w. p. } 9/64 \\ (y+13,y+3) & \text{w. p. } 3/64 \end{cases}$$

$$\Pr[X_{\mathcal{B}}(m)|X_{\mathcal{B}}(m-3)=\ell] = \begin{cases} (y+16,y) & \text{w. p. } 1/64 \\ (y+12,y+4) & \text{w. p. } 12/64 \\ (y+9,y+7) & \text{w. p. } 18/64 \\ (y+8,y+8) & \text{w. p. } 27/64 \\ (y+11,y+5) & \text{w. p. } 3/64 \\ (y+13,y+3) & \text{w. p. } 3/64 \end{cases}$$

Figure 3.1: Enumerating all the cases of both allocations

Let $\Gamma(X(m-3))$ denote the set of outcomes of random variable $X(m-3)$. We get,

$$\begin{aligned} E[L_{\mathcal{A}}] &= E[E[L_{\mathcal{A}}|X(m-3)]] \\ &= \sum_{\ell\in\Gamma(X(m-3))} \Pr[X(m-3)=\ell]\cdot E[L_{\mathcal{A}}|X(m-3)=\ell] \\ &= \sum_{\ell\in\Gamma(X(m-3)),d(\ell)=3} \Pr[X(m-3)=\ell]\cdot E[L_{\mathcal{A}}|X(m-3)=\ell] + \\ &\quad \sum_{\ell\in\Gamma(X(m-3)),d(\ell)\geq 6} \Pr[X(m-3)=\ell]\cdot E[L_{\mathcal{A}}|X(m-3)=\ell] \\ &> \sum_{\ell\in\Gamma(X(m-3)),d(\ell)=3} \Pr[X(m-3)=\ell]\cdot E[L_{\mathcal{B}}|X(m-3)=\ell] + \\ &\quad \sum_{\ell\in\Gamma(X(m-3)),d(\ell)\geq 6} \Pr[X(m-3)=\ell]\cdot E[L_{\mathcal{B}}|X(m-3)=\ell] \\ &= E[L_{\mathcal{B}}]. \end{aligned}$$

The third inequality is due to Equation 3.6 and 3.7. □

Remark Observation 3.4.20 shows that the decreasing order does *not* always give us the smallest expected maximum load when $n = 2$. However, we have not been able to generalize the result to $n \geq 3$. We feel that the decreasing order always yields the smallest expected maximum load when we have considerable number of bins. Our empirical study in Section 3.4.5 provides some evidence to support this argument.

3.4.5 Many Small Balls

Another natural question to ask is the one we answered in Corollary 3.4.10 for the single-choice game. Is it better to allocate a large number of small balls compared to a smaller number of large balls with the same total weight? The next example shows again that the majorization relation is not always maintained in the multiple-choice game.

Observation 3.4.21 *Let us consider two systems $\mathcal{A}$ and $\mathcal{B}$ both of n bins. Denote $W_{\mathcal{A}} = (0, 2, 4, \ldots, 2^{m-1})$ and $W_{\mathcal{B}} = (1, 1, 4, \ldots, 2^{m-1})$ to be two allocations both of $m \geq 3$ balls. Note both systems are of same total weight and $W_{\mathcal{A}} \succ W_{\mathcal{B}}$, but if m is odd, the expected maximum load of $\mathcal{A}$ is smaller than $\mathcal{B}$.*

Proof. Clearly after allocating the first two balls System $\mathcal{A}$ majorizes System $\mathcal{B}$. Besides, note that for both systems, the weight of every newly allocated ball is bigger than the sum of weights of all the balls allocated before. Hence, by Lemma 3.4.18, every time when a new ball is allocated, the majorization relation would be "reversed". Hence, for any odd number $m \geq 3$, System $\mathcal{B}$ certainly majorizes System $\mathcal{A}$. □

To see this, when $m = 3$, simply by enumerating all cases we can get, the expected maximum load of $\mathcal{A}$ is $4 + 2/n^2$, which is smaller than that of $\mathcal{B}$ $(4 + 4/n^2 - 2/n^4)$.

We emphasize again that the initial majorization relation is no longer preserved during the allocation. However, we still conjecture that in "most" cases the allocation of a large number of small balls is majorized by the one of a smaller number of large balls with the same total weight. Furthermore, it seems that in all cases, the expected maximum loads of the two allocations at most differ by an additive factor of the maximum ball size. Unfortunately, so far we have been unable to prove formal results. The next section contains empirical results obtained by computer simulations examining some of the issue we have raised earlier.

Simulation Results

In this section we conduct an empirical study for the weighted multiple-choice balls-into-bins game. We allocate $m = n$ balls into n bins while the number of choices, d, is chosen to be 2. We examine cases in which n is set to 100, 200, 500, and 1000. Note it is not feasible to enumerate the huge number of possible allocations (which is $n^{m \cdot d}$) to calculate the exact expected maximum loads. Instead, we approximate them by taking the average maximum loads for a large number (specifically 100,000) number of iterations.

The goal of the first experiment is to demonstrate the following observation: the more balanced the ball weights are, the less the expected maximum load will be, after allocating all balls. In our experiment, we first randomly assign a weight in $(0, 1)$ to each ball. After that, we perform a few "mixing" steps, in which we choose two balls at random and equalize their weights, to make the overall weight vectors more balanced. We record the corresponding expected maximum loads vs. the number of mixing steps in Figure 3.4.5.

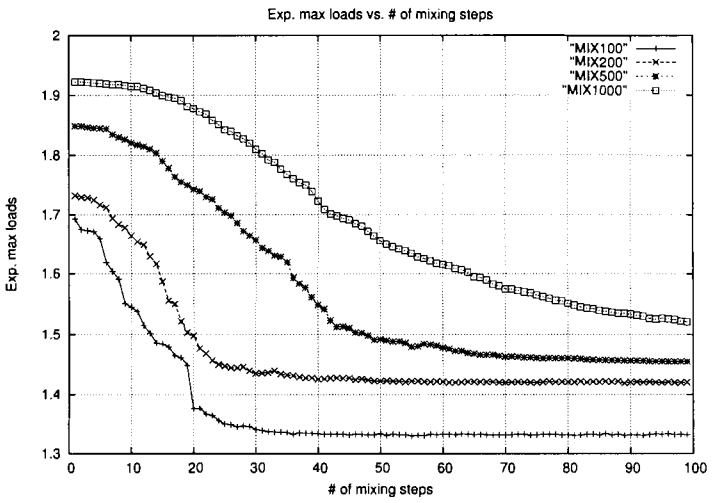

Figure 3.2: Successive equalization of weights

Although the first observation above is almost always true, we still note that there do exist ball weight distributions which achieve smaller expected maximum load than their corresponding uniform ones, as shown in Theorem 3.4.17.

Next we perform an experiment regarding the order of placing balls. We aim at showing that if we allocate balls in decreasing order of their weights, we would get the least expected maximum load. This seems intuitively likely since if we allocate big balls first, the small balls later are likely to fall into the holes left by the big ones. For the experiment, we first randomly assign each ball a weight in $(0, 1)$ and sort all ball weights by non-increasing order. Later, we perform a number of "swaps", i.e., we randomly choose two balls and exchange their weights, to get different ball arrangements. Figure 3.4.5 shows the relation between the number of swaps and the corresponding expected maximum loads.

Clearly our experiment appears to support the conjecture that the decreasing order achieves the minimum expected maximum load when n is large (Recall that counterexamples

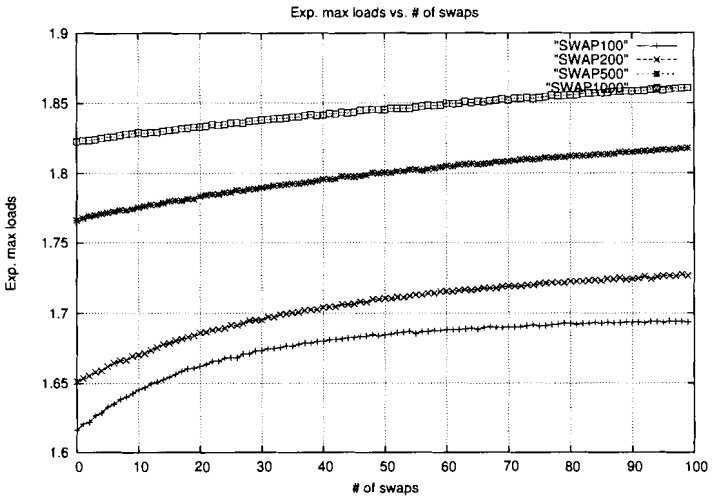

Figure 3.3: Successive swapping of weights

do exist when $n = 2$, see Observation 3.4.20). Unfortunately, we have not yet succeeded in proving this conjecture.

3.5 Selfish Reallocation Game

In this section we consider the problem of dynamically reallocating (or re-routing) m balls among a set of n bins (one may think of the balls as selfish users). Initially every bin is associated with some balls. Then in each round, every ball applies a natural, distributed algorithm (Algorithm 4) to reallocate itself into a different bin. Using game theoretic notion, when all the balls stop moving the system reaches some Nash equilibrium (or some state close to Nash equilibrium). We shall first introduce the notion of Nash equilibrium and its variations.

Nash Equilibrium

The status of an allocation is represented by a vector $X(t) = (x_1(t), \ldots, x_n(t))$ in which $x_i(t)$ denotes the *load* of bin i at the end of step t, i.e., the sum of weights of balls allocated to bin i. We will *normalize* load vector $X(t)$ by assuming a non-increasing order of bin loads, i.e. $x_1(t) \geq x_2(t) \geq \cdots \geq x_n(t)$. For any ball $b \in [m]$, let $r_b(t)$ denote the current bin of ball b at step t. In the following, we shall drop "t" if it is clear from the content.

Definition. [*Nash equilibrium*] An assignment is a *Nash equilibrium* for ball b if

$$x_{r_b} \leq x_j + w_b \text{ for all } j \in [n], \tag{3.8}$$

i.e., if ball b cannot improve its situation by migrating to any other bin.

Definition. [*ϵ-Nash equilibrium*] For $1 \geq \epsilon \geq 0$, we say a state is an *ϵ-Nash equilibrium* for ball b if

$$x_{r_b} \leq x_j + (1+\epsilon)w_b. \tag{3.9}$$

Notice that this definition is somewhat different from (and stronger than), e.g. Chien and Sinclair's in [35] where they say that (translated into our model) a state is an ϵ'-Nash equilibrium for $\epsilon' \in (0,1)$ if $(1-\epsilon')x_{r_b} \leq x_j + w_b$ for all $j \in [n]$. However, our definition captures theirs: for $\epsilon' \in (0,1)$ let $\epsilon = \frac{1}{1-\epsilon'} - 1 (> 0)$ and observe that $x_{r_b} \leq x_j + (1+\epsilon)w_b \leq (1+\epsilon)(x_j + w_b) = (1 + (\frac{1}{1-\epsilon'} - 1))(x_j + w_b) = \frac{x_j + w_b}{1-\epsilon'}$.

3.5.1 Weighted Case

We define our allocation process for weighted balls and uniform bins. $X_1(0), \ldots, X_n(0)$ is the initial assignment. The transition from state $X(t) = (x_1(t), \ldots, x_n(t))$ to state $X(t+1)$ is given by the algorithm below. Let $0 \leq \epsilon \leq 1$ and $\rho = \epsilon/8$.

Algorithm 4 Greedy Reallocation Protocol for Weighted Tasks

1: **for** each ball b in parallel **do**
2: let r_b be the current bin of ball b
3: choose bin j uniformly at random
4: **if** $X_{r_b}(t) \geq X_j(t) + (1+\epsilon)w_b$ *//violation of Equation 3.9//* **then**
5: move ball b from bin r_b to j with probability $\rho\left(1 - \frac{X_j(t)}{X_{r_b}(t)}\right)$

If the process converges, i.e. if $X(t) = X(t+1)$ for all $t \geq \tau$ for some $\tau \in \mathbb{N}$, then the system reached some ϵ-Nash equilibrium ("some" because ϵ-Nash equilibria are, in general, not unique). Our goal is to bound the number of steps it takes for the algorithm to converge, that is, to find the smallest τ with the property from above. We prove the following convergence result.

Theorem 3.5.1 *Let $\epsilon > 0$ and $\rho = \epsilon/8$. Let $\Delta \geq 1$ denote the maximum weight of any task. Let T be the number of rounds taken by the protocol in Figure 4 to reach an ϵ-Nash*

equilibrium for the first time. Then,

$$E[T] = O(mn\Delta^3(\rho\epsilon)^{-1}).$$

Preliminary Results

In this section we give some necessary notation for the analysis and prove some preliminary results. We use a standard potential function (see also [22]).

$$\Phi(x) = \sum_{i=1}^{n}(x_i - \overline{x})^2 = \sum_{i=1}^{n} x_i^2 - n\overline{x}^2 \tag{3.10}$$

In the following we assume, without loss of generality, that the assignment is "normalized", meaning $x_1 \geq \cdots \geq x_n$. If it is clear from the context we will omit the time parameter t in $X(t) = (X_1(t), \ldots, X_n(t))$ and write $X = (X_1, \ldots, X_n)$ instead. We say ball b has an *incentive* to move to bin i if $x_{r_b} \geq x_i + (1+\epsilon)w_b$ (notice that this is the condition used in line 4 of Algorithm 4).

Let $Y^b = (Y^b(r_b, 1), \ldots, Y^b(r_b, n))$ be a random variable with $\sum_{i=1}^{n} Y^b(r_b, i) = 1$. Y^b is an n-dimensional unit vector with precisely one coordinate equal to 1 and all others equal to 0. $Y^b(r_b, i) = 1$ corresponds to the event of ball b moving from bin r_b to bin i (or staying at bin i if $i = r_b$). Let the corresponding probabilities $(P^b(r_b, 1), \ldots, P^b(r_b, n))$ be given by

$$P^b(r_b, i) = \begin{cases} \frac{\rho(1-x_i/x_{r_b})}{n} & \text{if } r_b \neq i \text{ and } x_{r_b} > x_i + (1+\epsilon)w_b \\ 0 & \text{if } r_b \neq i \text{ and } x_{r_b} \leq x_i + (1+\epsilon)w_b \\ 1 - \sum_{k \in [n]: x_i > x_k} P^b(i,k) & \text{if } r_b = i. \end{cases}$$

The first (second) case corresponds to randomly choosing bin i and finding (not finding) an incentive to migrate, and the third case corresponds to randomly choosing the current bin.

For $i \in [n]$, let $S_i(t)$ denote the set of balls currently on bin i at step t. In the following we will omit t in S_i and write S_i if it is clear from the content. For $i, j \in [n]$ with $i \neq j$, let $I_{j,i}$ be the total weight of balls on bin j that have an incentive to move to bin i, i.e.,

$$I_{j,i} = \sum_{b \in S_j:\ x_j \geq x_i + (1+\epsilon)w_b} w_b.$$

Let

$$E[W_{j,i}] = \sum_{b \in S_j} w_b \cdot P^b(j,i) \leq x_j \cdot \frac{\rho(1 - x_i/x_j)}{n} = \frac{\rho(x_j - x_i)}{n}$$

denote the expected total weight of balls migrating from bin j to bin i (the last inequality is true because $I_{j,i} \leq x_j$; at most all the balls currently on j migrate to i). Next, we show three simple observations.

Observation 3.5.2

1. $\Phi(x) = \frac{1}{n} \cdot \sum_{i=1}^{n} \sum_{k=i+1}^{n} (x_i - x_k)^2$.
2. $E[\Phi(X(t+1))|X(t) = x] = \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x] - \overline{x}\Big)^2 + \sum_{i=1}^{n} \text{var}[X_i(t+1)|X(t) = x]$.
3. $\Phi(x) \leq m^2 \Delta^2$.

Proof. Part (1) is similar to Lemma 10 in [23]. To prove Part (2), by definition of Φ we have

$$
\begin{aligned}
& E[\Phi(X(t+1))|X(t) = x] \\
= & \sum_{i=1}^{n} E[X_i(t+1)^2|X(t) = x] - n\overline{x}^2 \\
= & \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x]^2 + \text{var}[X_i(t+1)|X(t) = x]\Big) - n\overline{x}^2 \\
= & \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x]^2 - \overline{x}^2\Big) + \sum_{i=1}^{n} \text{var}[X_i(t+1)|X(t) = x] \\
= & \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x] - \overline{x}\Big)^2 + 2\overline{x} \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x] - \overline{x}\Big) \\
& + \sum_{i=1}^{n} \text{var}[X_i(t+1)|X(t) = x] \\
= & \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x] - \overline{x}\Big)^2 + \sum_{i=1}^{n} \text{var}[X_i(t+1)|X(t) = x].
\end{aligned}
$$

Notice that $\sum_{i=1}^{n} E[X_i(t+1)|X(t) = x] = n\overline{x}$ and thus $2\overline{x} \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t) = x] - \overline{x}\Big) = 0$. For Part (3), simply check the worst case that all the m balls are in one bin. □

3.5.2 Convergence to Nash Equilibrium

In this section we bound the number of time steps for the system to reach some Nash equilibrium. We first bound the expected potential change during a fixed time step t (Lemma 3.5.5).

We shall first prove two technical lemmas: bounds for $\sum_{i=1}^{n}(E[X_i(t+1)|X(t)=x]-\overline{x})^2$ and $\sum_{i=1}^{n}\mathrm{var}[X_i(t+1)|X(t)=x]$, respectively (Lemma 3.5.3 and Lemma 3.5.4).

Lemma 3.5.3

1. $\sum_{i=1}^{n}\left(E[X_i(t+1)|X(t)=x]-\overline{x}\right)^2 < \Phi(x)-(2-4\rho)\sum_{i=1}^{n}\sum_{k=i+1}^{n}E[W_{i,k}](x_i-x_k)$.
2. $\sum_{i=1}^{n}\left(E[X_i(t+1)|X(t)=x]-\overline{x}\right)^2 > \Phi(x)-2\sum_{i=1}^{n}\sum_{k=i+1}^{n}E[W_{i,k}](x_i-x_k)$.

Proof. Since $E[W_{i,j}]$ is the expected total weight migrating from bin i to j, we have $E[X_i(t+1)|X(t)=x] = x_i+\sum_{j=1}^{i-1}E[W_{j,i}]-\sum_{k=i+1}^{n}E[W_{i,k}]$; recall that we assume $x_1 \geq \cdots \geq x_n$. To estimate $\sum_{i=1}^{n}(E[X_i(t+1)|X(t)=x]-\overline{x})^2$, we use an indirect approach by first analyzing a (deterministic) load balancing process. We then use the load balancing process to show our desired result (see [23]).

We consider the following load balancing scenario. Assume that there are n bins and every pair of bins is connected so that we have a complete network. Initially, every resource $1 \leq i \leq n$ has $z_i = x_i$ balls on it. Assume that $z_1 \geq \ldots \geq z_n$. Then every pair of bins $(i,k), i<k$ concurrently exchanges $\ell_{i,k} = E[W_{i,k}] \leq \rho(x_i-x_k)/n = \rho(z_i-z_k)/n$ balls. If $i \geq k$ we assume $\ell_{i,k}=0$. Note that the above system is similar to one step of the diffusion load balancing algorithm on a complete graph K_n. In both cases the exact potential change is hard to calculate due to the concurrent load transfers. The idea we use now is to first "sequentialize" the load transfers, measure the potential drop after each of these sub-steps, and then to use these results to get a bound on the total potential drop for the whole step.

In the following we assume that every edge $e_s=(i,k), i,k \in [n], k>i$ is labeled with weight $\ell_{i,k} \geq 0$. Note that $\ell_{i,k}=0$ if $x_i \leq x_k$. Let $N=n(n-1)/2$ and $E=\{e_1,e_2,\ldots e_N\}$ be the set of edges sorted in increasing order of their labels. We assume the edges are sequentially activated, starting with the edge e_1 with the smallest weight. Let $z^s=(z_1^s,\ldots,z_n^s)$ be the load vector resulting after the activation of the first s edges. Note that $z^0=(z_1^0,\ldots,z_n^0)$ is the load vector before load balancing and $z^N=(z_1^N,\ldots,z_n^N)$ is the load vector resulting after the activation of all edges. Note that $\Phi(z^0)=\Phi(x)$ since $i \in [n]$, $z_i^0=z_i=x_i$. Moreover, by the definition of our load balancing process and since $\ell_{i,k}=E[W_{i,k}]$ we have

$$z_i^N = z_i+\sum_{j=1}^{i-1}\ell_{j,i}-\sum_{k=i+1}^{n}\ell_{i,k} = x_i+\sum_{j=1}^{i-1}E[W_{i,j}]-\sum_{k=i+1}^{n}E[W_{i,k}] = E[X_i(t+1)|X(t)=x]$$

Hence

$$\Phi(z^N) = \sum_{i=1}^{n} \left(z_i^N - \overline{x}\right)^2 = \sum_{i=1}^{n} \Big(E[X_i(t+1)|X(t)=x] - \overline{x}\Big)^2.$$

Next we bound $\Phi(z^N)$. For any $s \in [N]$, let $\Delta_s(\Phi) = \Phi(z^{s-1}) - \Phi(z^s)$ be the potential drop due to the activation of edge $e_s = (i,k)$. Note that

$$\Phi(z^0) - \Phi(z^N) = \sum_{s=1}^{N} \big(\Phi(z^{s-1}) - \Phi(z^s)\big) = \sum_{e_s \in E} \Delta_s(\Phi).$$

Now we bound $\Delta_s(\Phi)$. Since all edges are activated in increasing order of their weights we get $l_{i,j} \leq \ell_{i,k} = \rho(z_i - z_k)/n$ for any node j that is considered before the activation of e_s. Node i has $n-2$ additional neighbours, hence the expected load that it can send to these neighbours before the activation of edge $e_s = (i,k)$ is at most $(n-2)\ell_{i,k} < \rho(z_i - z_k) - \ell_{i,k}$. This gives us

$$z_i^{s-1} \geq z_i - (n-2)\ell_{i,k} > z_i - \rho(z_i - z_k) + \ell_{i,k}.$$

Similarly, the expected load that k receives before the activation of edge $e_s = (i,k)$ is at most $\rho(z_i - z_k) - \ell_{i,k}$. Hence,

$$z_k^{s-1} < z_k + \rho(z_i - z_k) - \ell_{i,k}.$$

Thus,

$$\begin{aligned} \Delta_s(\Phi) &= (z_i^{s-1})^2 + (z_k^{s-1})^2 - (z_i^{s-1} - \ell_{i,k})^2 - (z_k^{s-1} + \ell_{i,k})^2 \\ &= 2\ell_{i,k} \cdot (z_i^{s-1} - z_k^{s-1} - \ell_{i,k}) > (2-4\rho)\ell_{i,k}(z_i - z_k). \end{aligned}$$

Similarly, since $z_i^{s-1} < z_i$ and $z_k^{s-1} > z_k$, we get

$$\Delta_s(\Phi) = 2\ell_{i,k} \cdot (z_i^{s-1} - z_k^{s-1} - \ell_{i,k}) < 2\ell_{i,k}(z_i - z_k).$$

Next we bound $\Phi(z^N)$.

$$\begin{aligned} \Phi(z^0) - \sum_{i=1}^{n} \sum_{k=i+1}^{n} 2\ell_{i,k}(z_i - z_k) &< \Phi(z^N) = \Phi(z^0) - \sum_{e_s \in E} \Delta_s(\Phi) \\ &< \Phi(z^0) - \sum_{i=1}^{n} \sum_{k=i+1}^{n} (2-4\rho)\ell_{i,k}(z_i - z_k). \end{aligned}$$

Consequently, we get

$$\Phi(z^N) = \sum_{i=1}^{n} \Big(E[X_i(t+1)|X_i = x] - \overline{x}\Big)^2 < \Phi(x) - (2-4\rho) \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k),$$

and

$$\Phi(z^N) = \sum_{i=1}^{n} \Big(E[X_i(t+1)|X_i = x] - \overline{x} \Big)^2 > \Phi(x) - 2 \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k).$$

□

Lemma 3.5.4 $\sum_{i=1}^{n} \mathrm{var}[X_i(t+1)|X(t) = x] < (2-\epsilon) \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k)$.

Proof. First of all, note that $\{Y^b(r_b, i)\}$ and $\{Y^{b'}(r_{b'}, i)\}$ are independent for $b \neq b'$. Let $S_i(t)$ be the set of balls that is assigned to resource i in step t.

$$\begin{aligned}
&\mathrm{var}[X_i(t+1)|X(t) = x] \\
&= \mathrm{var}\left[\sum_b w_b \cdot Y^b(r_b, i) \right] = \sum_b w_b^2 \cdot \mathrm{var}[Y^b(r_b, i)] \\
&= \sum_{j=1}^{n} \sum_{b \in S_j(t)} w_b^2 \cdot \mathrm{var}[Y^b(r_b, i)] \\
&= \sum_{j \neq i} \sum_{b \in S_j(t)} w_b^2 \cdot P^b(r_b, i)(1 - P^b(r_b, i)) + \sum_{b \in S_i(t)} w_b^2 \cdot P^b(r_b, i)(1 - P^b(r_b, i)) \\
&< \sum_{j \neq i} \sum_{b \in S_j(t)} w_b^2 \cdot P^b(r_b, i) + \sum_{b \in S_i(t)} w_b^2 \cdot (1 - P^b(r_b, i)) \\
&= \sum_{j \neq i} \sum_{b \in S_j(t)} w_b^2 \cdot P^b(r_b, i) + \sum_{b \in S_i(t)} w_b^2 \cdot \sum_{j \neq i} P^b(r_b, j) \\
&= \sum_{j \neq i} \sum_{b \in S_j(t)} w_b^2 \cdot P^b(r_b, i) + \sum_{j \neq i} \sum_{b \in S_i(t)} w_b^2 \cdot P^b(r_b, j) \\
&\leq \sum_{j \neq i} \sum_{b \in S_j(t)} w_b \cdot \frac{x_j - x_i}{1+\epsilon} \cdot P^b(r_b, i) + \sum_{j \neq i} \sum_{b \in S_i(t)} w_b \cdot \frac{x_i - x_j}{1+\epsilon} \cdot P^b(r_b, j) \\
&= \sum_{j \neq i} E[W_{j,i}] \cdot \frac{x_j - x_i}{1+\epsilon} + \sum_{j \neq i} E[W_{i,j}] \cdot \frac{x_i - x_j}{1+\epsilon}
\end{aligned}$$

The second inequality holds since $(x_j - x_i) \geq (1+\epsilon) \cdot w_b$ whenever a ball b in bin j have an incentive to move to bin i (see Algorithm 4). Now note that $E[W_{i,j}] = 0$ whenever $x_j > x_i$. Hence,

$$\begin{aligned}
&\sum_{i=1}^{n} \mathrm{var}[X_i(t+1)|X(t) = x] \\
&= \sum_{i=1}^{n} \sum_{j : x_j > x_i} E[W_{j,i}] \cdot \frac{(x_j - x_i)}{1+\epsilon} + \sum_{i=1}^{n} \sum_{j : x_i > x_j} E[W_{i,j}] \cdot \frac{(x_i - x_j)}{1+\epsilon}
\end{aligned}$$

$$= \sum_{i=1}^{n}\sum_{j=i+1}^{n} 2E[W_{i,i}]\cdot\frac{(x_i-x_j)}{1+\epsilon} < (2-\epsilon)\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i-x_k).$$

□

Now we are ready to show the following lemma bounding the potential change during step t.

Lemma 3.5.5

1. $E[\Phi(X(t+1))|X(t)=x] < \Phi(x) - \frac{\epsilon}{2}\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i-x_k)$.
2. $E[\Phi(X(t+1))|X(t)=x] > \Phi(x) - \epsilon\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i-x_k)$.

Proof. To prove part (1), combining Observation 3.5.2(2), Lemma 3.5.3(1) and 3.5.4, we get

$$\begin{aligned}
&E[\Phi(X(t+1))|X(t)=x]\\
&= \sum_{i=1}^{n}\Big(E[X_i(t+1)|X(t)=x]-\overline{x}\Big)^2 + \sum_{i=1}^{n}\mathrm{var}[X_i(t+1)|X(t)=x]\\
&< \Phi(x) - (2-4\rho)\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i-x_k) + (2-\epsilon)\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i-x_k)\\
&= \Phi(x) - \frac{\epsilon}{2}\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i-x_k),
\end{aligned}$$

since $\rho = \epsilon/8$. The proof of part (2) is similar. □

It is easy to prove the following corollary.

Corollary 3.5.6 $\forall t \geq 0$, $E[\Phi(X(t+1))] < E(\Phi[X(t)])$.

Proof. By Lemma 3.5.5(1),

$$E[X(t+1)] = \sum_{x\in\Omega(X(t))} E[\Phi(X(t+1))|X(t)=x] < \sum_{x\in\Omega(X(t))}\Phi(x) = E[\Phi(X(t))].$$

□

Next we first show that if $\Phi(x) \geq 4n\Delta^2$, then the system potential decreases by a multiplicative factor of at least $\rho\epsilon/4$ per round expectedly (Lemma 3.5.7). We then show that whenever x is not ϵ-Nash equilibrium, every round the system potential decreases at least by an additive factor of $\rho\epsilon/(6m\Delta)$ in expectation (Lemma 3.5.9). With these two Lemmas, we are ready to show our main result (Theorem 3.5.1).

Lemma 3.5.7 *If $\Phi(x) \geq 4n\Delta^2$, Δ is the maximum ball weight. We have $E[\Phi(X(t+1))|X(t) = x] < (1 - \rho\epsilon/4)\Phi(x)$.*

Proof. We first bound $\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k)$. Recall that $E[W_{i,k}] = I_{i,k} \cdot (\rho(1 - x_k/x_i))/n$, where $0 \leq I_{i,k} \leq x_i$ is the total weight of balls in x_i which have an incentive to migrate to x_k. To prove our bound we only add up the cases when $I_{i,k} = x_i$. Note that if $I_{i,k} < x_i$, we have $x_i - x_k < (1+\epsilon)\Delta$, since otherwise every ball in bin i would have an incentive to move to bin k resulting in $I_{i,k} = x_i$.

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k) \\
&= \sum_{i=1}^{n}\sum_{k=i+1}^{n} I_{i,k} \cdot \frac{\rho(1 - x_k/x_i)}{n} \cdot (x_i - x_k) \\
&\geq \frac{\rho}{n} \cdot \sum_{i=1}^{n}\sum_{k:I_{i,k}=x_i} I_{i,k}(1 - x_k/x_i)(x_i - x_k) = \frac{\rho}{n} \cdot \sum_{i=1}^{n}\sum_{k:I_{i,k}=x_i} (x_i - x_k)^2 \\
&= \frac{\rho}{n} \cdot \left(\sum_{i=1}^{n}\sum_{k=i+1}^{n} (x_i - x_k)^2 - \sum_{i=1}^{n}\sum_{k:I_{i,k}<x_i} (x_i - x_k)^2\right) \\
&\geq \frac{\rho}{n} \cdot \left(n\Phi(x) - \binom{n}{2} \cdot (1+\epsilon)^2\Delta^2\right) \geq \frac{\rho\Phi(x)}{2},
\end{aligned}$$

since $\Phi(x) \geq 4n\Delta^2$ and $\epsilon \leq 1$. Now, using Lemma 3.5.5(1) we obtain,

$$E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - \frac{\epsilon}{2}\sum_{i=1}^{n}\sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k) \leq (1 - \rho\epsilon/4) \cdot \Phi(x).$$

□

It is easy to derive the following corollary from Lemma 3.5.7.

Corollary 3.5.8 *For any $t > 0$, $E[\Phi(X(t+1))] \leq \max\{8n\Delta^2, (1 - \rho\epsilon/8) \cdot E[\Phi(X(t))]\}$.*

Proof. We consider two cases for different values of $E[\Phi(X(t)]$. If $E[\Phi(X(t))] \leq 8n\Delta^2$, by Corollary 3.5.6, $E[\Phi(X(t+1))] \leq E[\Phi(X(t))] \leq 8n\Delta^2$. Next we show if $E[\Phi(X(t))] \geq 8n\Delta^2$, $E[\Phi(X(t+1))] \leq (1-\rho\epsilon/8) \cdot E[\Phi(X(t))]$. Let $A = \{x \in \Omega(X(t)) | \Phi(x) \leq 4n\Delta^2\}$, $B = \Omega(x) \setminus A$. Note that by definition $\sum_{x\in A} \Pr[X(t) = x] \cdot \Phi(x) \leq 4n\Delta^2$.

$$
\begin{aligned}
& E[\Phi(X(t+1))] \\
&= \sum_{x\in\Omega(X(t))} \{E[\Phi(X(t+1)|X(t) = x] \cdot \Pr[X(t) = x]\} \\
&\leq \sum_{x\in A} \{\Phi(x) \cdot \Pr[X(t) = x]\} + \sum_{x\in B} \{(1-\rho\epsilon/4)\Phi(x) \cdot \Pr[X(t) = x]\} \\
&\leq \sum_{x\in A} \{(1-\rho\epsilon/8)\Phi(x) \cdot \Pr[X(t) = x]\} + \sum_{x\in B} \{(1-\rho\epsilon/8)\Phi(x) \cdot \Pr[X(t) = x]\} \\
&= (1-\rho\epsilon/8) \cdot \sum_{x\in\Omega(X(t))} \{\Phi(x) \cdot \Pr[X(t) = x]\} = (1-\rho\epsilon/8) \cdot E[\Phi(X(t))]\}.
\end{aligned}
$$

The first inequality is due to Lemma 3.5.5(1) and Lemma 3.5.7. To show the second inequality, observe that

$$\sum_{x\in B} \Pr[X(t) = x] \cdot \Phi(x) \geq \sum_{x\in A} \Pr[X(t) = x] \cdot \Phi(x),$$

since

$$E[\Phi(X(t))] = \sum_{x\in A} \Pr[X(t) = x] \cdot \Phi(x) + \sum_{x\in B} \Pr[X(t) = x] \cdot \Phi(x) \geq 8n\Delta^2$$

and

$$\sum_{x\in A} \Pr[X(t) = x] \cdot \Phi(x) \leq 4n\Delta^2.$$

□

Next we show Lemma 3.5.9, which indicates that whenever the system is not at some ϵ-Nash equilibrium, the system potential decreases by an amount of $\rho\epsilon/(6m\Delta)$ in expectation during that step.

Lemma 3.5.9 *Assume that at step t the system is not at some ϵ-Nash equilibrium. We have $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - \frac{\rho\epsilon}{6m\Delta}$.*

Proof. We consider two cases for different values of x_1, the maximum load of a bin.

1. $x_1 > \overline{x} + 2\Delta$. In this case we have $x_1 > x_n + 2\Delta > x_n + (1+\epsilon)\Delta$ since $x_n \leq \overline{x}$ and $0 < \epsilon < 1$. Thus, every ball in bin 1 has an incentive to move to bin n. Using Lemma 3.5.5(1), we get

$$\begin{aligned} E[\Phi(X(t+1))|X(t)=x] &\leq \Phi(x) - \frac{\epsilon}{2} \cdot \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k) \\ &\leq \Phi(x) - \frac{\epsilon}{2} \cdot E[W_{1,n}](x_1 - x_n) \\ &= \Phi(x) - \frac{\epsilon}{2} \cdot \frac{x_1 \cdot \rho(1 - x_n/x_1)}{n} \cdot (x_1 - x_n) \\ &= \Phi(x) - \frac{\rho\epsilon(x_1 - x_n)^2}{2n} \\ &< \Phi(x) - \frac{2\rho\epsilon\Delta^2}{n} < \Phi(x) - \frac{\rho\epsilon}{6m\Delta}. \end{aligned}$$

2. $x_1 \leq \overline{x} + 2\Delta$. Since x is not ϵ-Nash equilibrium, there must be at least one ball b that has an incentive to migrate to some bin $v \neq r_b$. Note that $x_{r_b} - x_v \geq (1+\epsilon)w_b > 1$ and $x_{r_b} \leq x_1 \leq \overline{x} + 2\Delta$. Similar to Case 1,

$$\begin{aligned} E[\Phi(X(t+1))|X(t)=x] &\leq \Phi(x) - \frac{\epsilon}{2} \cdot E[W_{r_b,v}](x_{r_b} - x_v) \\ &\leq \Phi(x) - \frac{\epsilon}{2} \cdot w_b \cdot \frac{\rho(1 - x_v/x_{r_b})}{n} \cdot (x_{r_b} - x_v) \\ &= \Phi(x) - \frac{\rho \cdot \epsilon \cdot w_b \cdot (x_{r_b} - x_v)^2}{2x_{r_b} \cdot n} \\ &\leq \Phi(x) - \frac{\rho\epsilon}{2(\overline{x} + 2\Delta)n} \leq \Phi(x) - \frac{\rho\epsilon}{6m\Delta}. \end{aligned}$$

   For the last inequality, we use $\overline{x} \cdot n = W \leq m \cdot \Delta$ and $m \geq n$.

□

Proof of Theorem 3.5.1 We first show that after $\tau = 16(\epsilon\rho)^{-1} \log m$ steps, $E[\Phi(X(\tau))] \leq 8n\Delta^2$. By Observation 3.5.2(3), $\Phi(X(0)) \leq m^2\Delta^2$. Repeatedly using Corollary 3.5.8 we get $E[\Phi(X(\tau))] \leq \max\{8n\Delta^2, (1 - \rho\epsilon/8)^\tau \cdot \Phi(X(0))\} = 8n\Delta^2$. By Markov inequality, $\Pr[\Phi(X(\tau)) > 80n\Delta^2] \leq 0.1$.

The following proof is done by a standard martingale argument similar to [22] and [83]. Let us assume that $\Phi(X(\tau)) \leq 80n\Delta^2$. Let T be the number of additional time steps for the system to reach some ϵ-Nash equilibrium after step τ and let $t \wedge T$ be the minimum of

t and T. Let $V = \rho\epsilon/(6m\Delta)$ and let $Z_t = \Phi(X(t+\tau)) + Vt$. Observe that $\{Z_t\}_{t\wedge T}$ is a supermartingale since by Lemma 3.5.9 with $X(t+\tau) = x$,

$$E[Z_{t+1}|Z_t = z] = E\left[\Phi(X(\tau+t+1)) \mid \Phi(x) = z - Vt\right] + V\cdot(t+1) \leq (z - Vt - V) + V\cdot(t+1) = z.$$

Hence $E[Z_{t+1}] = \sum_z E[Z_{t+1}|Z_t = z] \cdot \Pr[Z_t = z] \leq \sum_z z \cdot \Pr[Z_t = z] = E[Z_t]$. We obtain

$$V \cdot E[T] \leq E[\Phi(X(\tau+T))] + V \cdot E[T] = E[Z_T] \leq \ldots \leq E[Z_0] \leq 80n\Delta^2.$$

Therefore $E[T] \leq 80n\Delta^2/V = 480mn\Delta^3(\rho\epsilon)^{-1}$, and $\Pr[T > 4800mn\Delta^3(\rho\epsilon)^{-1}] < 0.1$ by Markov's inequality. Hence, after $\tau + T = 16(\rho\epsilon)^{-1}\log m + 4800mn\Delta^3(\rho\epsilon)^{-1}$ rounds, the probability that the system is not at some ϵ-Nash equilibrium is at most $0.1 + 0.1 = 0.2$.

Subdivide time into intervals of $\tau + T$ steps each. The probability that the process has *not* reached an ϵ-Nash equilibrium after s intervals is at most $(1/5)^s$. This finishes the proof. □

Corollary 3.5.10 *Assume that every ball has integer weight of at least 1. After running Algorithm 4 with $\epsilon = 1/\Delta$ for $O(mn\Delta^5)$ steps, the chance that the system is not at some Nash equilibrium is at most* 0.2.

Proof. When Algorithm 4 terminates, for any ball b and bin $i \in [n]$, we have $x_{r_b} < x_i + (1+\epsilon)w_b < x_i + w_b + 1 \leq x_i + w_b$ since $w_b \leq \Delta$ and w_b is an integer. This implies that the system is at one of the Nash equilibria. Now, setting $\epsilon = 1/\Delta$ in Theorem 3.5.1 and using $\rho = \epsilon/8 = (8\Delta)^{-1}$, we obtain the result. □

Lower bound for the Convergence time

We prove the following lower bound result for the convergence time of Algorithm 4.

Observation 3.5.11 *Let T be the first time at which $X(t)$ is the Nash equilibrium. There is a load configuration $X(0)$ that requires $E[T] = \Omega(m\Delta/\epsilon)$.*

Proof. Consider a system with n bins and n uniform balls and $m - n$ balls with weight $\Delta \geq 2$. Let $l = m/n$ where m is a multiple of n. Let $X(0) = (\,(l-1)\Delta + 2, (l-1)\Delta + 1, \ldots, (l-1)\Delta + 1, (l-1)\Delta\,)$. The perfectly balanced state is the only Nash equilibrium. Let q be the probability for the unit-size balls in bin 1 to move to bin n (if exactly one of the two unit-sized balls moves, the system reaches the Nash equilibrium). By Algorithm 4,

we have $q = \rho \cdot 2/(n((l-1)\Delta + 2))) = O(\epsilon/m\Delta)$ since $l = m/n$ and $\rho = \epsilon/8$. Note that T is geometric distributed with probability $2q(1-q)$. Thus $E[T] = 1/(2q(1-q)) = \Omega(m\Delta/\epsilon)$. □

Remark We believe that since there is lack of global knowledge and also balls query the load of only one other server, even with significant change to the algorithm we can not omit the term Δ. For an evidence consider two different games as follow, both with two servers.

- There are 4 balls with weights $(1, 1, \Delta, \Delta)$ and the initial configuration is $(\Delta + 2, \Delta)$.
- There are $2\Delta + 2$ balls all of unit weight, and the initial configuration is $(\Delta + 2, \Delta)$.

Considering the lack of global knowledge, a ball with unit weight can not distinguish between the above games. But in order to have a fast convergence to the Nash Equilibrium (see [60] for the definition), in the first game it needs to migrate with a probability significantly higher than the corresponding probability in the second game.

3.5.3 Uniform Case

In this section we show convergence for Algorithm 4 for the case that all balls are uniform (i.e., $\Delta = 1$). [22] shows that the perfectly balanced state is the unique Nash equilibrium. We set $\epsilon = 1$ in Algorithm 4, thus $\rho = 1/(8\epsilon) = 1/8$.

Convergence to Nash Equilibrium

Note that when Algorithm 4 terminates, we have $\forall i, j \in n$, $x_i < x_j + (1+\epsilon)\Delta = x_j + 2$. Hence the system is in the Nash equilibrium. In the following, we show (in Theorem 3.5.16) that, after $O(\log m + n \log n)$ steps, Algorithm 4 terminates with high probability. This improves the previous upper bound of $O(\log \log m + n^4)$ in [22] for small values of m. In fact, we can actually combine these two algorithms to obtain a tight convergence time of $O(\log \log m + n \log n)$ w.h.p. The tightness of this result can be shown by Theorem 4.2 in [22] and Observation 3.5.18.

For simplicity we assume that m is a multiple of n, the proof can easily be extended to $n \nmid m$. We first prove Lemma 3.5.13, which is a similar result to Lemma 3.5.5(1) that bounds the expected potential drop in one round. Then we show that in each round the potential drops at least by a factor of $1/32$ if the current system potential is larger than

n (Lemma 3.5.15(1)), and at least by a factor of $1/8n$ otherwise (Lemma 3.5.15(2)). With these two lemmas, we are ready to show Theorem 3.5.16.

We will use the same potential function $\Phi(x)$ as the one in Section 3.5.1. Recall that by Observation 3.5.2(1), for an arbitrary load configuration x,

$$\Phi(x) = \sum_{i=1}^{n} (x_i - \overline{x})^2 = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=i+1}^{n} (x_i - x_k)^2.$$

For bin $i, k \in [n]$, let $E[W_{i,k}]$ denote the expected number of balls being transferred from bin i to k. Note that by Algorithm 4, if $x_i - x_k \geq 2$, $E[W_{i,k}] = x_i \cdot \rho(1 - x_k/x_i)/n = \rho(x_i - x_k)/n$, otherwise $E[W_{i,k}] = 0$. Let $S_i(x) = \{k : x_i \geq x_k + 2\}$ and $E_i(x) = \{k : x_i = x_k + 1\}$. Let

$$\Gamma(x) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in L_i(x)} (x_i - x_k)^2.$$

Note that the bigger $\Gamma(x)$ is, the more balls are expected to be transferred by Algorithm 4. We first show some relations between $\Gamma(x)$ and $\Phi(x)$.

Observation 3.5.12 *For any load configuration x, we have*

1. *If $\Phi(x) \geq n$, then $\Gamma(x) > \Phi(x)/2$.*
2. *If $\Gamma(x) < 2$, then $\Gamma(x) = \Phi(x)^2/n$ and $\Phi(x) < \sqrt{2n}$.*
3. *If $\Phi(x) < 2$, then x is Nash equilibrium.*

Proof. For Part (1), by definition,

$$\Gamma(x) = \Phi(x) - \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in E_i(x)} 1 \geq \Phi(x) - \frac{1}{n} \cdot \frac{n(n-1)}{2} = \Phi(x) - \frac{n-1}{2}.$$

Hence if $\Phi(x) \geq n$, we get $\Gamma(x) \geq \Phi(x) - (n-1)/2 > \Phi(x)/2$.

For Part (2), we first show that if $\Gamma(x) < 2$, then $x_1 - x_n \leq 2$ (notice that $x_1 \geq x_2 \geq \ldots, \geq x_n$). For a contradiction assume that $\Gamma(x) < 2$ and $x_1 - x_n \geq 3$. Hence $\forall 1 \leq i \leq n$, either $|x_i - x_1| \geq 2$ or $|x_i - x_n| \geq 2$. Also notice that by symmetry we have

$$\Gamma(x) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in S_i(x)} (x_i - x_k)^2 = \frac{1}{2n} \sum_{i=1}^{n} \sum_{k \in S_i(x)} (x_i - x_k)^2 \geq \frac{1}{2n} \cdot n \cdot (2^2) = 2,$$

resulting a contradiction.

Next we show $\Gamma(x) = \Phi(x)^2/n$. Since $\overline{x} = m/n$ is an integer and $x_1 - x_n \leq 2$, each bin can only have $\overline{x} - 1$, $\overline{x}$, $\overline{x} + 1$ balls. Let $A(B)$ be the set of bins with $\overline{x} - 1$ balls and ($\overline{x} + 1$ balls), respectively. Of course $|A| = |B| : r$. Thus $\Phi(x) = \sum_{i=1}^{n} (x_i - \overline{x})^2 = 2r$. Hence,

$$\Gamma(x) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in S_i(x)} (x_i - x_k)^2 = \frac{1}{n} \cdot (2r)^2 = 4r^2/n = \Phi(x)^2/n.$$

Consequently, given $\Gamma(x) < 2$, we have $\Phi(x) \leq \sqrt{2n}$.

For Part (3), for a contradiction assume that x is not Nash equilibrium. Then there must be two bins u, v, such that $x_u \geq \overline{x} + 1$ and $x_v \leq \overline{x} - 1$. Thus $\Phi(x) = \sum_{i=1}^{n} (x_i - \overline{x})^2 \geq 2$. We get a contradiction.

□

We then show the following bound for the expected potential drop in one step.

Lemma 3.5.13 $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - \Gamma(x)/16$.

Proof. Recall that if $x_i - x_k \geq 2$, $E[W_{i,k}] = \rho(x_i - x_k)/n$, otherwise $E[W_{i,k}] = 0$. Hence,

$$\Gamma(x) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k \in S_i(x)} (x_i - x_k)^2 = \rho^{-1} \cdot \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k).$$

Setting $\epsilon = 1$ and $\rho = 1/8$ in Lemma 3.5.5(1), we get

$$E[\Phi(X(t+1))|X(t) = x] = \Phi(x) - \frac{1}{2} \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k) = \Phi(x) - \frac{\rho\Gamma(x)}{2} = \Phi(x) - \frac{\Gamma(x)}{16}.$$

□

The following corollaries follow from Lemma 3.5.13.

Corollary 3.5.14

1. *If* $\Phi(x) \geq n$, $E[\Phi(X(t+1))|X(t) = x] < (1 - 1/32)\Phi(x)$.
2. *If* $n > \Phi(x) \geq \sqrt{2n}$, $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - 1/8$.
3. *If* $\sqrt{2n} > \Phi(x)$, $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - \Phi(x)^2/(16n)$.

Proof. Part (1) follows directly from Lemma 3.5.13 and Observation 3.5.12(1).

To prove Part (2), if $\Phi(x) \geq \sqrt{2n}$, by Observation 3.5.12(2) $\Gamma(x) \geq 2$. Then use Lemma 3.5.13 we get $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - 1/8$.

For Part (3), note that $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - \Gamma(x)/16$ by Lemma 3.5.13. Thus it is sufficient to show that $\Gamma(x) \geq \Phi(x)^2/n$. We consider two cases for different values of $\Gamma(x)$. If $\Gamma(x) \geq 2$, $\Gamma(x) > \Phi(x)^2/n$ since $\Phi(x) < \sqrt{2n}$. If $\Gamma(x) < 2$, by Observation 3.5.12(2), $\Gamma(x) = \Phi(x)^2/n$. □

Next we prove two results that bound the expected potential drop.

Lemma 3.5.15 *For any $t > 0$,*

1. $E[\Phi(X(t+1))] \leq \max\{n, (1 - 1/32)E[\Phi(X(t))]\}$.
2. $E[\Phi(X(t+1))] \leq \left(1 - \frac{1}{8n}\right) E[\Phi(X(t))]$.

Proof. The proof of Part (1) is similar to Corollary 3.5.8. If $E[\Phi(X(t))] \leq 2n$, by Corollary 3.5.6, $E[\Phi(X(t+1))] \leq E[\Phi(X(t))] \leq 2n$. In the following we show if $E[\Phi(X(t))] \geq 2n$, $E[\Phi(X(t+1))] \leq (1 - 1/64)E[\Phi(X(t))]$. Let $A = \{x \in \Omega(X(t))|\Phi(x) \leq n\}$ and $B = \Omega(X(t)) \setminus A$. Note that by definition $\sum_{x \in A} \Pr[X(t) = x] \cdot \Phi(x) \leq n$.

$$
\begin{aligned}
& E[\Phi(X(t+1))] \\
&= \sum_{x \in \Omega(X(t))} E[\Phi(X(t+1)|X(t) = x] \cdot \Pr[X(t) = x] \\
&\leq \sum_{x \in A} \{\Phi(x) \cdot \Pr[X(t) = x]\} + \sum_{x \in B} \{(1 - 1/32)\Phi(x) \cdot \Pr[X(t) = x]\} \\
&\leq \sum_{x \in A} \{(1 - 1/64)\Phi(x) \cdot \Pr[X(t) = x]\} + \sum_{x \in B} \{(1 - 1/64)\Phi(x) \cdot \Pr[X(t) = x]\} \\
&= (1 - 1/64) \cdot \sum_{x \in \Omega(X(t))} \{\Phi(x) \cdot \Pr[X(t) = x]\} = (1 - 1/64) \cdot E[\Phi(X(t))].
\end{aligned}
$$

The first inequality is due to Lemma 3.5.5 and Corollary 3.5.14(1). To show the second inequality, observe that

$$\sum_{x \in B} \Pr[X(t) = x] \cdot \Phi(x) \geq \sum_{x \in A} \Pr[X(t) = x] \cdot \Phi(x),$$

since

$$E[\Phi(X(t))] = \sum_{x \in A} \Pr[X(t) = x] \cdot \Phi(x) + \sum_{x \in B} \Pr[X(t) = x] \cdot \Phi(x) \geq 2n$$

and

$$\sum_{x \in A} \Pr[X(t) = x] \cdot \Phi(x) \leq n.$$

To prove Part (2), we first show that for any load configuration x, $E[\Phi(X(t+1))|X(t) = x] \leq (1 - 1/(8n))\Phi(x)$. There are four cases for different values of $\Phi(x)$.

1. If $\Phi(x) \geq n$, by Corollary 3.5.14(1), $E[\Phi(X(t+1))|X(t) = x] < (1 - 1/32)\Phi(x) < (1 - 1/(8n))\Phi(x)$ as long as $n > 4$.
2. If $n > \Phi(x) \geq \sqrt{2n}$, by Corollary 3.5.14(2), $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - 1/8 \leq (1 - 1/(8n))\Phi(x)$ since $\Phi(x)/(8n) < 1/8$ due to $\Phi(x) < n$.
3. If $\sqrt{2n} > \Phi(x) \geq 2$, by Corollary 3.5.14(3), $E[\Phi(X(t+1))|X(t) = x] \leq \Phi(x) - \Phi(x)^2/(16n) \leq (1 - 1/(8n))\Phi(x)$ since $\Phi(x) \geq 2$.
4. Finally, if $\Phi(x) < 2$, by Observation 3.5.12(3), x must be Nash equilibrium so that $\Phi(x) = 0$. In this case the system potential will not change. Hence $E[\Phi(X(t+1))|X(t) = x] = 0 \leq (1 - 1/(8n))\Phi(x)$.

Consequently,

$$\begin{aligned} E[\Phi(X(t+1))] &= \sum_x E[\Phi(X(t+1)|X(t) = x] \cdot \Pr[X(t) = x] \\ &\leq \sum_x \left(1 - \frac{1}{8n}\right) \Phi(x) \cdot \Pr[X(t) = x] = \left(1 - \frac{1}{8n}\right) E[\Phi(x)]. \end{aligned}$$

□

Theorem 3.5.16 *Given any initial load configuration $X(0) = x$. The probability that the system does not reach the Nash equilibrium after $64 \log m + 16n \ln n$ steps is at most $1/n$.*

Proof. We first show that after $\tau = 128 \ln m$ steps, $E[\Phi(X(\tau))] \leq n$. By Observation 3.5.2(3), $\Phi(X(0)) \leq m^2\Delta^2 = m^2$. Using Lemma 3.5.15(1) iteratively for τ times, we get

$$E[\Phi(X(\tau))] \leq \max\{2n, (1 - 1/64)^\tau \cdot \Phi(X(0))\} \leq \max\{n, (1 - 1/64)^\tau \cdot m^2\} = 2n.$$

We then show that after $T = 24n \ln n$ additional steps, the system reaches Nash equilibrium w.h.p. Using Lemma 3.5.15(2) iteratively for T times, we get

$$E[\Phi(X(\tau+T)] \leq E[\Phi(X(\tau))] \cdot (1 - 1/(8n))^T \leq (2n) \cdot (1 - 1/(8n))^{24n \ln n} < n \cdot e^{-3 \ln n} = n^{-1}.$$

By Markov's inequality, $\Pr[\Phi(X(\tau+T)) \geq 2] < 1/n$. Observation 3.5.12(3) tells us that if $\Phi(X(\tau+T)) < 2$, $X(\tau+T)$ is Nash equilibrium. Hence, after $\tau + T = 128 \ln m + 24n \ln n$ steps, the probability that the system does not reach the Nash equilibrium is at most $1/n$. □

Remark Note that we can combine Algorithm 4 and Algorithm 1 in [22] to obtain an algorithm that converges in $O(\log\log m + n \log n)$ steps. To see this, first note that by Corollary 3.9 in [22], after $T_1 = 2 \log\log m$ steps, $E[\Phi(X(T_1)] \leq 18n$. Then using a similar argument as above, we can show that after $O(\log\log m + n \log n)$, the system state is at some Nash equilibrium w.h.p.

Lower bounds

We prove the following two lower bound results which show the tightness of Theorem 3.5.16.

Observation 3.5.17 *Let T be the first time at which $E[X(t)] \leq c$ for constant $c > 0$. There is an initial load configuration $X(0)$ that requires $T = \Omega(\log m)$.*

Proof. Consider a system with $n = 2$ bins and m uniform balls. Let $X(0) = (m, 0)$. We first show that $E[\Phi(X(t+1)] \geq \frac{7}{8} E[\Phi(X(t))]$. By definition,

$$\sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k) = \frac{\rho}{n} \left(\sum_{i=1}^{n} \sum_{k=i+1}^{n} I_{i,k}(1 - x_k/x_i)(x_i - x_k) \right) \leq \frac{\Phi(x)}{8}.$$

Hence, setting $\epsilon = 1$ in Lemma 3.5.5(2) we obtain

$$E[\Phi(X(t+1))|X(t) = x] \geq \Phi(x) - \epsilon \sum_{i=1}^{n} \sum_{k=i+1}^{n} E[W_{i,k}](x_i - x_k) \geq \frac{7\Phi(x)}{8}.$$

Now similar to Lemma 3.5.15 (1) we can show that $E[\Phi(X(t+1))] \geq 7E[\Phi(X(t))]/8$. Note that $\Phi(X(0)) = m^2/2$. In order to make $E[X(T)] \leq c$, we need $T = \Omega(\log m)$. □

Observation 3.5.18 *Let T be the first time at which $X(t)$ is a Nash equilibrium and T^* be the upper bound for T. There is an initial load configuration $X(0)$ that in order to make $\Pr[T \leq T^*] > 1 - 1/n$, we need $T^* = \Omega(n \log n)$.*

Proof. Consider a system with n bins and m uniform balls. Let $X(0)$ be the assignment given by $X(0) = (2, 1, \ldots, 1, 0)$. Denote q be the probability for the balls in bin 1 to

move to bin n (if exactly one of the two balls in bin 1 moves, the system reaches the Nash equilibrium). By Algorithm 4 (with $\rho = 1/8$), $q = 2/(2\rho n) = 1/(8n)$. Note that T is geometric distributed with probability $2q(1-q) < 1/(4n)$. Consequently, $\Pr[T > T^*] \leq (1/(4n))^{T^*}$ (since step $1, \ldots, T^*$ all must fail). Thus, to have $\Pr[T \leq T^*] > 1 - 1/n$, we need $T^* = \Omega(n \log n)$. □

Remark Note that this lower bound also holds for the algorithm in [22] (with $\rho = 1$).

3.6 Summary

In this chapter we have studied the weighted balls-into-bins games where every ball is associated with some positive weight. we have considered two different scenarios, the *static sequential game* and the *selfish reallocation game.*

Static Sequential Game In the static sequential game, balls arrive without initial locations. Our goal is to allocate them into bins as evenly as possible. We have studied a well-known approach that to have every ball choose $d \geq 1$ bins independently and uniformly at random, and allocate itself into the bin with the lightest load. We have shown that for the single-choice game, i.e., $d = 1$, a more balanced weight distribution always yields a smaller expected maximum load. Our proof is based on the majorization technique. For the multiple-choice game, we first showed that the expected maximum load is not necessarily minimized for the allocation with uniform balls when we have sufficiently many balls. We then proved that the majorization order is not generally preserved. Regarding the order in which we allocate balls, we proved that the expected maximum load is not necessarily maximized if we allocate balls in increasing order. We then showed that the decreasing order does not always yield the smallest expected maximum load when $n = 2$.

There are two interesting open questions in the static sequential game. First, it would be interesting to generalize Theorem 3.4.17 to allow arbitrary number of balls. Second, we proved that the decreasing order does not necessarily yield the smallest expected maximum load when $n = 2$. We are interested in the question whether the counterexample still holds for arbitrary n.

Selfish Reallocation Game In the selfish reallocation game, every bin is initially associated with some balls. Then each ball applies the following natural, distributed reallocation

algorithm to reallocate itself into different bin. In each round, every ball first picks a bin uniformly at random. It then compares the load of its current host bin with the load of the randomly chosen bin. If the load difference is above a certain threshold then the ball will migrate to the destination bin with a certain probability. Our goal is to bound the convergence time, which is the number of steps for the system to terminate.

For the weighted case where each ball $i \in [m]$ is associated with some weight $w_i \geq 1$, We proved that after $O(mn\Delta^3\epsilon^{-2})$ steps, the system converges to the ϵ-Nash equilibrium with probability at least $4/5$, where Δ is the maximum task weight. Our analysis is based on the potential function technique. We also proved a lower bound of $\Omega(m\Delta/\epsilon)$ for the convergence time. For the uniform case where every ball has uniform weight (i.e., $\Delta = 1$), we proved that the system converges to the (real) Nash equilibrium in $O(\log m + n \log n)$ steps w.h.p. We also obtained an algorithm with tight convergence time of $O(\log \log m + n \log n)$ by combining our algorithm with the one in [22].

The first open question in the selfish reallocation game is whether we can close the gap between the upper and lower bounds for the weighted case. Yet, we believe that to answer this question, a different potential function is necessary. Next, instead of the linear latency function used in this work, we can consider more general latency functions, e.g., functions with "bounded jump" property in [35] or "bounded relative slope" property in [61]. As a first step, we can assume that every bin is associated with some positive "speed". The latency of a bin would then be the load of that bin divided by the speed.

Chapter 4

Energy Efficient Routing in Ad Hoc Networks

In this chapter we study how to design efficient routing algorithms for broadcasting and gossiping in ad hoc networks. Our goal is not only to minimize the broadcasting and gossiping time, but also to minimize the *energy consumption*, which is measured in terms of the total *number of messages* (or *transmissions*) sent. We consider ad hoc networks with both random and general topologies.

For random networks, we present a broadcasting algorithm where every node transmits at most once. We show that our algorithm broadcasts in $O(\log n)$ time steps (rounds), w.h.p., where n is the number of nodes. We then present a $O(d \log n)$ (d is the expected degree) gossiping algorithm using $O(\log n)$ messages per node. For general networks with known diameter D, we present a randomized broadcasting algorithm with optimal broadcasting time $O(D \log(n/D) + \log^2 n)$ that uses an expected number of $O(\log^2 n / \log(n/D))$ transmissions per node. We also show a trade-off result between the broadcasting time and the number of transmissions: we construct a network such that any oblivious algorithm using a time-invariant distribution requires $\Omega(\log^2 n / \log(n/D))$ messages per node in order to finish broadcasting in optimal time. This demonstrates the tightness of our upper bound. We also show that no oblivious algorithm can complete broadcasting w.h.p. using $o(\log n)$ messages per node.

4.1 Introduction

In this chapter we consider a typical disributed system, the ad hoc network. We study how to design efficient routing algorithms for two fundamental communication problems in ad hoc networks, *broadcasting* and *gossiping*. For broadcasting, one node sends a message to the rest of the network. For gossiping, every node sends a message to all other nodes in the network.

An ad hoc network consists of a set of mobile nodes connected through wireless links. The main advantage of an ad hoc network is that it does not need any infrastructure. Thus, ad hoc networks are easier to deploy and are more scalable than traditional networks. Due to these advantages, ad hoc networks have received much attention in recent years. In an ad hoc network, nodes model the wireless devices equipped with antennas. Every device has a fixed *communication range* and it can listen to all neighbouring devices within that range. We assume that all devices share only one communication channel. Hence for a fixed device, if several devices within its communication range transmit at the same time, these messages "collide" and the receiver is not able to receive any of them. Moreover, in an ad hoc network, nodes can have different communication ranges, one node might be able to listen to another, but not vice-versa. This forbids the acknowledgement based protocols, since nodes might not be able to send a confirmation message to the sender upon receiving a message. Another challenge is that, due to the mobility of wireless nodes, the topology of an ad hoc network can change rapidly and nonpredictably. Thus, it is commonly assumed that the topology of the network is unknown to the network nodes. Particularly, a node does not know which nodes are within its communication range, or even the number of neighbours. Hence, it is desirable that communication algorithms use local information only.

The communication problem in ad hoc networks has been extensively studied in the literature. See Section 4.2 for an overview. The major goal is to minimize the *broadcasting/gossiping time*, i.e., the number of rounds to achieve broadcasting/gossiping. Yet, since the mobile devices tend to be small and have only small batteries, another important issue for communication in ad hoc networks is energy efficiency (e.g., [66, 85]). In this work we design efficient communication algorithms which minimize both the broadcasting (gossiping) time and the energy consumption. Since we do not assume variable communication ranges, we can assume that devices cannot adjust the energy need for send operations. This allows us to measure the energy consumption in terms of the number of total transmissions.

We will also show that there is a tradeoff between minimizing the broadcasting or gossiping time, and the number of messages that are needed by randomized protocols.

The rest of this chapter is organized as follows. In Section 4.2 we introduce the related work. Section 4.3 introduces our model and new results. We study broadcasting and gossiping for random networks in Section 4.4 and Section 4.5, respectively. In Section 4.6, we propose and analyze a broadcasting algorithm on general (not random but fixed) networks with known diameter. Our algorithm minimizes both the broadcasting time and the number of transmissions. We also give some lower bound results on the number of transmissions.

4.2 Related Work

In the following we review broadcasting and gossiping algorithms(protocols) for unknown ad hoc networks. There are mainly two classes of approaches, *randomized* and *deterministic*. In each round of a randomized algorithm, all active nodes transmit with identical probability that is chosen according to some predetermined probability distribution. In each round of a deterministic algorithm, we specify which active nodes will transmit. Let D be the diameter of the network.

4.2.1 Randomized Broadcasting

Arbitrary Networks

Alon et al. [7] show that there exists a network with diameter $O(1)$ for which broadcasting takes expected time $\Omega(\log^2 n)$. Kushilevitz and Mansour [76] show a lower bound of $\Omega(D \log(n/D))$ time for any randomized broadcasting algorithm. Bar-Yehuda, Goldriech and Itai [14] design an almost optimal broadcasting algorithm which achieves the broadcasting time of $O((D + \log n) \log n)$, w.h.p.

Later, Czumaj and Rytter [51] propose an elegant algorithm which achieves (w.h.p.) linear broadcasting time on arbitrary networks. Their algorithm uses carefully defined selection sequences which specify the probabilities that are used by the nodes to determine if a message should be sent or not. This algorithm needs $\Theta(n)$ transmissions per node. Czumaj and Rytter [51] also obtain an algorithm under the assumption that the network diameter is known. The algorithm finishes broadcasting in $O(D \log(n/D) + \log^2 n)$ rounds,

w.h.p., and uses expected $\Theta(D)$ transmissions per node. Also, independently, Kowalski and Pelc [73] obtain a similar randomized algorithm with the same running time.

In the following, we review the linear time randomized broadcasting algorithm in [51] in more detail.

Algorithm 5 (The linear randomized broadcasting algorithm from [51])

for each round r **do**

Choose a transmitting probability I_r according to the following distribution:

$$\Pr[I_r = 2^{-k}] = \begin{cases} 2^{-(k+1)} & \text{for } 1 \leq k \leq \log\log n, \\ \frac{1}{2\log n} & \text{for } \log\log n < k \leq \log n, \\ 1 - \sum_{i=1}^{\log n} \Pr[I_r = 2^{-i}] & \text{for } k = 0. \end{cases}$$

Every informed node transmits with probability I_r.

Theorem 4.2.1 *(Theorem 1 from [51]) Algorithm 5 completes broadcasting in $O(n)$ rounds with probability at least $1 - n^{-1}$.*

We briefly sketch the proof of the broadcasting time in [51]. Let u be the originator of the broadcast and let v be an arbitrary node. Let T be the random variable representing the number of rounds before v is informed. Fix an arbitrary shortest path $P = \{u = u_1, \ldots u_{L+1} = v\}$ of length $L \leq D$ from u to v. Let $layer_P(i)$ denote the set such that $\forall w \in layer_P(i)$, u_i is the highest ranked node on the path P that w has an edge to. The set $layer_P(i)$ is called the layer of *rank* i with respect to P. Note that $\forall i, j, layer_P(i) \cap layer_P(j) = \phi$ and $|\sum_{i=1}^{n} layer_P(i)| \leq n$. We say $layer_P(i)$ is *leading* at Round r if $layer_P(i)$ is the highest ranked layer consisting of an informed node at Round r. Let T_i be the random variable representing the number of rounds that $layer_P(i)$ is leading. Note that $T = \sum_{i=1}^{n} T_i$.

It is shown in [51] that T_i follows geometric distribution with probability at least $\frac{1}{20\min\{\log n, |layer_P(i)|\}}$. Hence

$$E[T] = \sum_{i=1}^{n} E[T_i] \leq 20 \sum_{i=1}^{n} |layer_P(i)| \leq 20n.$$

Furthermore, [51] proves the following concentration result for the sum of a set of geometrically distributed random variables.

Lemma 4.2.2 *(Lemma 3.5 from [51]) Let $X_1, \ldots, X_n$ be a sequence of independent integer-valued random variables, each X_i being geometrically distributed with a parameter $p_i, 0 < p_i < 1$. For every i, $1 \leq i \leq \ell$, let $\mu_i = 1/p_i$ and assume that all μ_is are from a set Δ, that is $\Delta = \{\mu_i : 1 \leq i \leq \ell\}$. If $\sum_{i=1}^{\ell} \mu_i \leq N$, then for every positive real number β,*

$$\Pr\left[\sum_{i=1}^{\ell} X_i \leq 2N + 8\ln(|\Delta|/\beta) \cdot \sum_{z \in \Delta} z\right] \geq 1 - \beta.$$

Note that $E[T_i] \leq 20 \log n$. Lemma 4.2.2 with $\beta = n^{-2}, N = 20n$, we get

$$\Pr[T \geq 2 \cdot 20n + 8\ln(20n^2 \log n)(20 \log n)^2] \leq 1 - n^{-2}.$$

Finally, applying the Union bound we conclude that Algorithm 5 completes broadcasting in $O(n)$ time.

Using similar idea as above, Czumaj and Rytter [51] prove the following theorem for broadcasting on shallow networks (with known diameter D).

Theorem 4.2.3 *(Theorem 2 from [51]) Let N be a network with diameter D, there exists an algorithm that can complete broadcasting in $O(\log^2 n + D \log(n/D))$ rounds with probability at least $1 - n^{-1}$.*

Random Networks

Elsässer and Gasieniec [55] are the first to study the broadcasting problem on the class of directed random networks (graphs) $\mathbb{G}(n, p)$. In these networks, every pair of nodes is connected with probability p. They propose a randomized algorithm which achieves w.h.p. strict logarithmic broadcasting time. Their algorithm is as follows.

Algorithm 6 (The randomized broadcasting algorithm from [55])

Phase 1:

for Round 1 to $D - 1$ **do**

Every informed node transmits with probability 1.

Phase 2: (Round D)

Every informed node transmits with probability n/d^D.

Phase 3:

for Round $D + 1$ to $O(\log n)$ **do**

Every informed node transmits with probability $1/d$.

Elsässer and Gasieniec prove that Algorithm 6 finishes broadcasting in $O(\log n)$ steps w.h.p. Elsässer and Gasieniec [55] also propose a centralized algorithm that achieves $O(\ln n/\ln d + \ln d)$ running time. The authors also show that this algorithm is asymptotically optimal.

In [56], Elsässer studies the communication complexity of broadcasting in random networks under the so-called *random phone call* model, in which every node forwards its message to a randomly chosen neighbour at every round. The proposed algorithm can complete broadcasting in $O(\log n)$ rounds by using at most $O(n \max\{\log\log n, \log n/\log d\})$ transmissions, which is optimal under his random phone call model.

4.2.2 Deterministic Broadcasting

The problem of deterministic broadcasting is also extensively studied. For arbitrary networks, Chlebus et al. [36] give the first sub-quadratic algorithm with running time of $O(n^{11/6})$. Chrobak, Gasieniec and Rytter [40] propose the first $O(n \cdot poly(\log n))$ algorithm that can complete broadcasting in $O(n\log^2 n)$ rounds. Kowalski and Pelc [74] show that there exists a deterministic algorithm that can complete broadcasting in $O(n\log n\log D)$ rounds using a complicate non-constructive counting argument. Very recently, Czumaj and Rytter [51] obtain a deterministic algorithm with running time $O(\log^2 D)$, which is an extension of their proposed randomized algorithm. The best known lower bound $\Omega(n\log D)$ is due to Clementi, Monti and Silvestri [44].

4.2.3 Gossiping

For gossiping, all the previous work follows the join model, where nodes are allowed to join messages originated from different nodes together to one large message. Chrobak, Gasieniec and Rytter [41] propose a randomized gossiping algorithm that achieves $O(n\log^4 n)$ gossiping time. This result was improved to $O(n\log^3 n)$ by Liu and Probhakaran [81]. Czumaj and Rytter [51] obtain so far the fastest randomized algorithm that has a running time of $O(n\log^2 n)$. The algorithm combines the linear time broadcasting algorithm of [51], and a framework proposed by [41]. The framework applies a series of limited broadcasting phases (with broadcasting time $O(f(n))$) to do gossiping in time $O(\max\{n\log n, f(n)\log^2 n\})$. Chlebus, Kowalski and Rytter [39] study the average-time complexity of gossiping in ad hoc networks. They give a gossiping protocol that works in average time of $O(n/\log n)$, which

is shown to be optimal. For the case when k different nodes initiate broadcasting (note that it is gossiping when $k = n$), they give an algorithm with $O(\min\{k \log(n/k) + n/\log n\})$ average running time. So far, the fastest deterministic gossiping algorithm has a running time of $O(n^{1.5})$ [112].

4.2.4 Random Graphs

Finally, we review some basic topological properties of random graphs. In the classic random graph model of Erdös and Rényi, $\mathbb{G}(n,p)$ is a n-node graph where any pair of vertices is connected (i.e., an edge is built in between) with probability p. It can be shown by Chernoff bounds that every node in the network has $\Theta(d)$ neighbours w.h.p. Moreover, It is well-known (e.g. [31, 42]) that as long as $p = \Omega(\log n/n)$, the diameter of the graph is $(1 + o(1))(\log n/\log d)$ w.h.p. Besides, if $p > \log n/n$, the graph is connected w.h.p.

4.3 Model and New Results

An ad hoc network is modeled by a directed graph $G = (V, E)$. V is the set of devices and $|V| = n$. For $u, v \in V$, $(u, v) \in E$ means that u is in the communication range of v (but not necessarily vice versa). We assume that the network G is unknown, meaning that the nodes do not have any knowledge about the nodes that can receive their messages, nor the number of nodes from which they can receive messages by themselves. This assumption makes sense since in a lot of applications the graph G is not fixed because the mobile agents can move around (which will results in a changing communication structure). In order to make our problem feasible, we assume that our network is strongly connected, i.e., there is a path between any pair of nodes.

We assume that G is either arbitrary [7, 51, 76], or that it belongs to the random network class [55]. For random graphs, we use a directed version of the standard model $\mathbb{G}(n,p)$, where node v has an edge to node w with probability p. Let d be the average in and out degree of G. Recall that $d = np$ and $D = (1 + o(1))(\log n/\log d)$.

In the broadcasting problem one node of the network tries to send a message to all other nodes in the network, whereas in the case of gossiping every node of the network tries to sends a message to all other nodes. The *broadcasting time* (or the *gossiping time*) denotes the number of communication rounds needed to finish broadcasting (or gossiping). The *energy consumption* is measured in terms of the total (expected) number of transmissions, or the

maximum number of transmissions per node. The algorithms we consider are *oblivious*, i.e., all nodes have to use the same algorithm.

Broadcast in random networks Our broadcasting algorithm is similar to the one of Elsässer and Gasieniec in [55](also Algorithm 6). The difference is that our algorithm sends at most one message per node, whereas Algorithm 6 sends up to $D-1$ messages per node. The broadcasting time of both algorithms is $O(\log n)$ w.h.p. Our proof is very different from the one in [55]. Elsässer and Gasieniec show first some structural properties of random graphs which are used to analyze their algorithm. We directly bound the number of nodes which received the message after every round. Our results are also more general in the sense that we only need $p = \omega(\log n/n)$ instead of $p = \omega(\log^{\delta} n/n)$ for constant $\delta > 1$ (see [55]).

Gossiping in Random Networks We modify the algorithm of [51] and achieve a gossiping algorithm (Algorithm 8) with running time $O(d \log n)$ w.h.p., where every node sends only $O(\log n)$ messages. To our best knowledge, this is the first gossiping algorithm specialized on random networks. So far, the fastest gossiping algorithm for general network achieves $O(n \log^2 n)$ running time and uses an expected number of $O(n \log n)$ transmissions per node [51].

Broadcasting in General networks Our randomized broadcasting algorithm for general networks completes broadcasting time $O(D \log(n/D) + \log^2 n)$, w.h.p. It uses an expected number of $O(\log^2 n/\log(n/D))$ transmissions per node. Czumaj and Rytter ([51]) propose a randomized algorithm with $O(D \log(n/D) + \log^2 n)$ broadcasting time. Their algorithm can easily be transformed into an algorithm with the same runtime bounds and an expected number of $\Omega(\log^2 n)$ transmissions per node.

Lower Bounds for General networks First we show a lower bound of $n \log n/2$ transmissions for any randomized broadcasting algorithm with a success probability of at least $1 - n^{-1}$. We assume that every node in the network uses the same probability distribution to determine if it sends a message or not. Furthermore, we assume that the distribution does not change over time. To our best knowledge, all distributions used so far have these properties. Czumaj and Rytter ([51]) propose an algorithm that needs $O(n \log^2 n)$ messages (see Section 4.2). Hence, there is still a factor of $\log n$ messages left between upper and our lower bound.

Finally, using the same lower bound model, we show that there is a network with $O(n)$ nodes and diameter D, such that every randomized broadcast algorithm requires an expected number of at least $\log^2 n/(\max\{4c, 8\}\log(n/D))$ transmissions per node in order to finish broadcasting in $cD\log(n/D)$ rounds with probability at least $1 - n^{-1}$. This lower bound shows the optimality of our proposed broadcasting algorithm (Algorithm 9).

4.4 Broadcasting in Random Networks

In this section we present our broadcasting algorithm for random networks. Our algorithm is based on the algorithm proposed in [55] (See Algorithm 6 in Section 4.2.1). The algorithm completes broadcasting in $O(\log n)$ rounds w.h.p, which matches the result in [55].

Let $T = \lfloor \log n/\log d \rfloor$. Throughout the analysis, we always assume that $n = |V|$ is sufficiently large, and $p > \delta \log n/n$ for a sufficiently large constant δ. Note that the latter condition is necessary for the network to be connected w.h.p. In the following, every node that already got the message is called *informed*. An informed node v can be in one of two different states. v is called *active* as soon as it is informed, and it will become *passive* (meaning it will never transmit a message again) as soon as it tried once to send the message. The main idea of the algorithm is as follows.

Phase 1. The goal of Phase 1 is to inform $\Theta\left(d^T\right)$ nodes w.h.p. (Lemma 4.4.4). To prove this result, we repeatedly use Lemma 4.4.3, which bounds the number of active nodes after each round.

Phase 2. The goal of Phase 2 is to inform $\Theta(n)$ nodes w.h.p. when $p \leq n^{-2/5}$ (Lemma 4.4.5). For the other cases we do not need Phase 2.

Phase 3. The goal of Phase 3 is to inform every remaining uninformed node w.h.p. (Lemma 4.4.6).

We prove the following theorem.

Theorem 4.4.1 *If $p > \delta \log n/n$ for a sufficiently large constant δ, Algorithm 7 completes broadcasting in $O(\log n)$ rounds, w.h.p. Furthermore, every node performs at most one transmission and the expected total number of transmissions is $O(\log n/p)$.*

Algorithm 7 An Energy efficient algorithm for Random Networks

Phase 1:

1: The state of the source is set to *active*.
2: **for** round $r = 1$ to T **do**
3: Every *active* node v transmits once and becomes *passive*.
4: **if** node v receives the message *for the first time* **then**
5: The status of v is set to *active*.

Phase 2:

1: **if** $p \leq n^{-2/5}$ **then**
2: Every active node transmits with probability $1/(d^T p)$ and becomes *passive*.
3: **if** node v receives the message *for the first time* **then**
4: The status of v is set to *active*.

Phase 3:

1: **for** round $r = 0$ to $\beta \log n$ (β is a constant) **do**
2: **if** $p \leq n^{-2/5}$ **then**
3: Every active node transmits with probability $1/d$
4: A node that has transmitted the message becomes *passive*.
5: **else**
6: Every active node transmits with probability $1/(dp)$
7: A node that has transmitted the message becomes *passive*.

The number of transmissions performed in Phase 1 is $1 + d + \ldots + d^{T-1} = O(1/p)$ since $T = \lfloor \log n / \log d \rfloor$. The (expected) number of transmissions in each round of Phase 2 and 3 is bounded by $1/p$. Hence, the expected total number of transmissions is $O(\log n/p)$.

To proof Theorem 4.4.1 it remains to bound the broadcasting time. This part of the proof is split into several lemmas. Let U_t be the set of active nodes at the beginning of Round t, Q_t be the set of nodes which transmit in Round t. Let N_t be the number of uninformed nodes at the beginning of Round t. We first prove the following simple observations which will be used in the later sections.

Observation 4.4.2

1. $\forall t \in [1,T]$, $U_t = Q_t$.
2. $\forall t \in [1,T], N_t = n - \left(\sum_{i=1}^{t-1} |Q_i| + |U_t|\right)$.
3. $\forall r, t \geq 1$, $r < t$, $|U_t| \geq |U_r| - \sum_{i=r}^{t-1} |Q_i|$.
4. $Q_i \bigcap Q_j = \phi$ *for all* $i, j \geq 1$ *with* $i \neq j$.

Proof. (1) is true since in Phase 1 of our algorithm every active node transmits. To prove (2), note that for any informed node v at Round t, there are only two possibilities: either v transmits in some round between 1 and $t-1$ (i.e., $v \in Q_i, i \in [1, t-1]$), or v must be active at Round t, (i.e., $v \in U_t$). For (3), simply note that nodes being active in Round r will remain active until Round t if they do not transmit in the meantime. For (4), note that every node only transmits at most once per broadcast. □

Observation 4.4.2(4) helps us to argue that the random experiments used later in the analysis are independent from each other. In the following, we first prove Lemma 4.4.3 (1) showing that in each round of Phase 1 the number of active nodes grows by a factor of $\Theta(d)$, w.h.p. The second part of Lemma 4.4.3 strengthens the results if the number of active nodes is between $[\log^3 n, \frac{1}{p \log n}]$.

4.4.1 Analysis of Phase 1

Lemma 4.4.3 *If* $p > \delta \log n/n$ *and* $1 \leq t \leq T$ *(Phase 1), then the following statements are true with a probability* $1 - o(n^{-4})$.

1. *For* $0 < |U_t| < 1/p$, $(d/16)|U_t| < |U_{t+1}| < (2d)|U_t|$.

2. *For* $\log^3 n < |U_t| < 1/(p \log n)$, $(1 - 3/\log n)\, d|U_t| < |U_{t+1}| < (1 + 1/\log n)\, d|U_t|$.

Proof. We consider two cases of different values of p. If $p > 1/2$, we have $T = 1$ and every node will have expectedly $(n-1)/2$ neighbours. The result now follows from a simple application of Chernoff bounds. If $p \leq 1/2$, we fix an arbitrary node u and a round $t = 1$ in Phase 1. First we bound q, the probability that u is informed in Round t, i.e. u is connected to *exactly* one node in U_t.

$$q = |U_t|p(1-p)^{|U_t|-1} > p|U_t|(1-p)^{1/p} \geq p|U_t|/4. \tag{4.1}$$

Here, the first inequality uses the condition $|U_t| < 1/p$. To see the second one, note that $\forall 0 < p < 1/2, (1-p)^{1/p} > 1/4$. Next, we show N_t, the number of uninformed nodes at time t, is larger than $n/2$. By Observation 4.4.2(2),

$$\begin{aligned} N_t &= n - \left(\sum_{i=1}^{t-1} |Q_i| + |U_t| \right) > n - t|U_t| \\ &> n - (\log n)(1/p) > n/2. \end{aligned} \tag{4.2}$$

Here, the first inequality is true by Observation 4.4.2(1) and $|U_1| < |U_2| < \ldots < |U_t|$. The second one uses the condition $|U_t| < 1/p$ and $t \leq T = \lfloor \log n / \log d \rfloor \leq \log n$. The third inequality uses $p > \delta \log n/n$. Hence,

$$\begin{aligned} E[|U_{t+1}|] &= N_t q > (n/2) \cdot q \\ &\geq (n/2) \cdot p|U_t|/4 = d|U_t|/8, \end{aligned}$$

since $N_t > n/2$ and $d = np$. Note that the events to be connected to exactly one node in U_t are independent for different uninformed nodes. Also, note that each event is only evaluated once due to Observation 4.4.2(4). Using Chernoff bounds we get

$$\begin{aligned} \Pr[U_{t+1}| \leq d|U_t|/16] &\leq \Pr[|U_{t+1}| \leq E[|U_{t+1}|]/2] \\ &\leq e^{-\frac{d|U_t|}{64}} = o(n^{-4}). \end{aligned}$$

The last inequality uses $d = np$ with $p = \delta \log n/n$ for a sufficiently large constant δ. Consequently $|U_{t+1}|/|U_t| > d/16$ with a probability $1 - o(n^{-4})$. Using a similar approach, we can prove that $|U_{t+1}|/|U_t| < 2d$ with a probability $1 - o(n^{-4})$. This finished the proof of part 1 of the lemma.

To prove part 2 we first need a tighter bound on q. By Equation 4.1,

$$q = |U_t|p \cdot (1-p)^{|U_t|-1} > (1 - p|U_t|)p|U_t| > (1 - 1/\log n) \cdot p|U_t|$$

Next we bound N_t. Using Equation 4.2 with $|U_t| < 1/(p \log n)$ and $t \leq T = \lfloor \log n / \log d \rfloor \leq \log n$ we get

$$N_t = n - \left(\sum_{i=1}^{t-1} |Q_i| + |U_t| \right) > n - t|U_t| > n - 1/p > n\,(1 - 1/\log n)\,.$$

Now, we obtain the following lower bound for $E[|U_{t+1}|]$,

$$E[|U_{t+1}|] = N_t q > (1 - 1/\log n)^2\, d|U_t|\,(1 - 2/\log n)\, d|U_t|.$$

For an upper bound on $E[|U_{t+1}|]$ we use $N_t < n$ and $q \leq p|U_t|$ to get

$$E[|U_{t+1}|] = N_t q < np|U_t| = d|U_t|.$$

Using Chernoff bounds together with the assumption that $|U_t| > \log^3 n$, we get

$$\Pr\left[(1 - 3/\log n)\, d|U_t| < |U_{t+1}| < (1 + 1/\log n)\, d|U_t|\right] > 1 - 2e^{-E[|U_{t+1}|]/(3\log^2 n)} = 1 - o\left(n^{-4}\right).$$

□

Now, we are ready to show the following concentration result for $|U_{T+1}|$, the number of active nodes after Phase 1.

Lemma 4.4.4 *Let $c_1 = 16^{-4}4^{-3}$, and $c_2 = 16e$. After Phase 1 we have with a probability of $1 - o(n^{-3})$*

$$c_1 d^T \leq |U_{T+1}| \leq c_2 d^T.$$

Proof. By Observation 4.4.2(4), the random experiments performed in different rounds are independent from each other. Hence, we can repeatedly use Lemma 4.4.3 to bound $|U_{T+1}|$.

Case 1: $p \geq n^{-4/5}$ Since $d = np \geq n^{1/5}$, $T = \lfloor \log n / \log d \rfloor \leq 4$. Using Lemma 4.4.3(1) for T rounds, we get $(d/16)^T \leq |U_{T+1}| \leq (2d)^T$ with a probability $1 - o(n^{-3})$. To show that we can use Lemma 4.4.3(1) for Round $1 \leq i \leq T$, we note that $|U_i| \leq (2d)^{T-1} \leq 8d^{T-1} < 1/p$ since $T \leq 4$ and $d \geq \delta \log n/n$. The lemma now follows from the choices of c_1 and c_2.

Case 2: $n^{-4/5} > p > \delta \log n/n$ In this case we have $T = \lfloor \log n/\log d \rfloor \geq 5$. Using Lemma 4.4.3(1) for three rounds, we get $|U_4| \geq (d/16)^3 > \log^3 n$ w.h.p since $d = np > \delta \log n$. Again, we can use Lemma 4.4.3(1) for the first three rounds. After three rounds, the condition of Lemma 4.4.3(2) is w.h.p. fulfilled. In the following we show that $|U_i|$ does not increase too fast such that we are allowed to use Lemma 4.4.3(2) for Round $4 \leq i \leq T-1$, i.e. $\log^3 n < |U_i| < 1/(p \log n)$. For the first inequality, note that $|U_i|$ does not decrease for large values of i (Lemma 4.4.3(1)), w.h.p. For the second inequality we use Lemma 4.4.3(1) for the first three rounds and then Lemma 4.4.3(2) for the remaining $i-4$ rounds, we get

$$\begin{aligned} |U_i| &< (2d)^3 (1 + 1/\log n)^{i-4} d^{i-4} \\ &< 8 (1 + 1/\log n)^{\log n} d^{i-1} \\ &< (8e) d^{T-2} < 1/(p \log n). \end{aligned}$$

The first inequality uses the fact that $i < T = \lfloor \log n/\log d \rfloor \leq \log n$. The second inequality uses that $\forall 0 < x < 1, (1+x)^{1/x} < e$ and $i \leq T-1$. The last inequality holds because $d^{T-1} < 1/p$ by definition of T and $d = np > \delta \log n$. This shows that we can use Lemma 4.4.3(2) for Round $4 \leq i \leq T-1$. Similarly, we get

$$\begin{aligned} |U_T| &< (2d)^3 (1 + 1/\log n)^{T-4} d^{T-4} \\ &< 8 (1 + 1/\log n)^{\log n} d^{T-1} \\ &< (8e) d^{T-1} < 1/p, \end{aligned}$$

the last inequality holds by $T = \lfloor \log n/\log d \rfloor$. This shows that we can use Lemma 4.4.3(1) for Round T.

Now we are ready to bound $|U_{T+1}|$. We use Lemma 4.4.3(1) for three rounds, Lemma 4.4.3(2) for the next $T-4$ rounds, and then Lemma 4.4.3(1) once again. Now we applying the Union bound and get with a probability $1 - o(n^{-3})$

$$|U_{T+1}| \geq (d/16)^3 \cdot (d(1 - 3/\log n))^{T-4} \cdot (d/16),$$

and,

$$|U_{T+1}| \leq (2d)^3 \cdot (d(1 + 1/\log n))^{T-4} \cdot (2d).$$

Since $T \leq \log n$, and $\forall 0 \leq x \leq 1/2, (1-x)^{1/x} > 1/4$, we get

$$(d/16)^4 (d(1 - 3/\log n))^{T-4} > (1/16)^4 (1 - 3/\log n)^{\log n} d^T > (16^{-4} 4^{-3}) d^T.$$

Similarly, we get

$$(2d)^3 \left(d\left(1+1/\log n\right)\right)^{T-4} (2d) < 2^4 \left(1+1/\log n\right)^{\log n} d^T < (16e)d^T.$$

This shows that with a probability $1 - o(n^{-3})$ we have

$$(16^{-4}4^{-3})d^T \leq |U_{T+1}| \leq (16e)d^T.$$

□

4.4.2 Analysis of Phase 2

Next we show a result for Phase 2. If $n^{-2/5} > p > \delta \log n/n$ for a sufficiently large constant δ, Lemma 4.4.5 shows that after Phase 2 the number of active nodes is $\Theta(n)$, w.h.p. For the rest case we do not need Phase 2.

Lemma 4.4.5 *Let $c = c_1 4^{-2c_2-1}$. If $n^{-2/5} > p > \delta \log n/n$ for a sufficiently large constant δ, after Phase 2 (Round $T+1$) we have with a probability of $1 - o(n^{-3})$, $|U_{T+2}| > c\, n$.*

Proof. Phase 2 only consists of Round $T+1$ in which every active node transmits with probability $1/(d^T p)$. We first prove bounds for $|Q_{T+1}|$. By Lemma 4.4.4,

$$c_2/p > E[|Q_{T+1}|] = |U_{T+1}| \cdot 1/(d^T p) > c_1/p.$$

Using Chernoff bounds we get

$$\begin{aligned} \Pr[c_1/2p \leq |Q_{T+1}| \leq 2c_2/p] &> 1 - 2e^{-E[|Q_{T+1}|]/3} \\ &> 1 - 2e^{-(c_1/(3p))} = 1 - o(n^{-3}). \end{aligned} \tag{4.3}$$

Now we fix an arbitrary but uninformed node v. We show the probability to inform v in Phase 2 is constant. In order to inform v, v must be connect to exactly one node in Q_{T+1}. Hence, using Equation 4.3 together with the fact that $\forall 0 < x < 1/2, (1-x)^{1/x} > 1/4$, we get

$$\Pr[v \text{ is informed}] = |Q_{T+1}|p(1-p)^{|Q_{T+1}|-1} \geq |Q_{T+1}|p(1-p)^{2c_2/p} > c_1 4^{-2c_2}.$$

Next we show that $N_{T+1} \geq n/2$, w.h.p. First note that we can assume that $|U_{T+1}| < n/4$. Otherwise, the lemma is already fulfilled by Observation 4.4.2(3) and Equation 4.3. This

holds since $|U_{T+2}| \geq |U_{T+1}| - |Q_{T+1}| \geq n/4 - 2c_2/p > n/8$ ($p \geq \delta \log n/n$). Now, using Observation 4.4.2(2),

$$\begin{aligned} N_{T+1} &= n - \left(\sum_{i=1}^{T} |Q_i| + |U_{T+1}| \right) \\ &> n - T|U_T| - |U_{T+1}| \\ &> n - \log n/p - n/4 > n/2, \end{aligned}$$

with a probability $1 - o(n^{-3})$. The first equation follows since $\forall 1 \leq i \leq T, Q_i = U_i$ and by Lemma 4.4.3, $|U_1| < |U_2| < \ldots < |U_T|$. The second inequality holds since $|U_T| < 1/p$, $T \leq \log n$ and $|U_{T+1}| < n/4$. The third inequality follows since $p > \delta \log n/n$ for a sufficiently large constant δ.

Next we estimate the expected number of active nodes at the end of Phase 2.

$$E[|U_{T+2}|] = N_{T+1} \cdot \Pr[v \text{ is informed}] \geq \left(c_1 4^{-2c_2}/2\right) n.$$

Note that the events that different uninformed nodes are connected to exactly one node in U_{T+1} are independent from each other. Also, note that, due to Observation 4.4.2(4), each of these events is evaluated only once. Using Chernoff bounds we get

$$\begin{aligned} \Pr[|U_{T+2}| \leq (c_1 4^{-2c_2-1})n] &\leq \Pr[|U_{T+2}| \leq E[|U_{T+2}|]/2] \\ &\leq e^{-E[|U_{T+2}|]/8} = o(n^{-3}). \end{aligned}$$

□

4.4.3 Analysis of Phase 3

Next, we show that after running Phase 3 for $O(\log n)$ rounds, every node is informed w.h.p. Note that even at the end of Phase 3, we still have a considerable amount of active nodes because in each round of Phase 3, only a small number of active nodes will transmit and become passive afterwards.

Lemma 4.4.6 *After running Phase 3 for* $128 \log n/c$ *rounds, every node is informed with a probability of* $1 - o(n^{-1})$.

Proof. Let $k = 128 \log n/c$. Fix some uninformed node v and let $A_t(v)$ be the number of active neighbours of v at the beginning of Round t of Phase 3. For any $0 \leq t \leq k$, let $f_t(v)$

be the number of active neighbours of v that transmitted before Round t of Phase 3. Note that $A_t(v) = A_0(v) - f_t(v)$. Let $P_t(v)$ be the probability to inform node v in Round t. In the following we consider two cases for different values of p.

Case 1: $n^{-2/5} \geq p > \delta \log n/n$ for a sufficiently large constant δ. We first show that $A_0(v) = \Theta(d)$, w.h.p. Note that $A_0(v)$ is the number of neighbours of v that are activated in Phase 2. Since the probability that v is connected to any node in U_{T+2} (the set of nodes that are activated in Phase 2) is p, $E[A_0(v)] = |U_{T+2}|p > cnp = cd$ with a probability at least $1 - o(n^{-3})$ by Lemma 4.4.5. Using Chernoff bounds we get,

$$\begin{aligned} \Pr[A_0(v) < cd/2] &\leq \Pr[A_0(v) < E[A_0(v)]/2] \\ &\leq e^{-E[A_0(v)](1/2)^2/2} = o(n^{-3}). \end{aligned} \tag{4.4}$$

The last inequality holds since $E[A_0(v)] > cnp$ with $p > \delta \log n/n$ for a sufficiently large constant δ. Similarly, we can show that

$$\Pr[A_0(v) \geq 2d] = o(n^{-3}). \tag{4.5}$$

Since every active neighbour of v transmits with probability $1/d$ in each round of Phase 3, we get

$$E[f_t(v)] \leq tA_0(v)/d \leq A_0(v)/(4e),$$

because $t \leq k = 128 \log n/c$ and $d = np$ with $p > \delta \log n/n$ for a sufficiently large constant δ. Using $\Pr[\mathbb{B}(n,p) > anp] < (e/a)^{anp}$ we get,

$$\Pr[f_t(v) > A_0(v)/2] \leq (e/(2e))^{A_0(v)/2} = o(n^{-3}).$$

The last inequality follows since by Equation 4.4, $A_0(v) > cd/2 > 6 \log n$. Consequently, it follows by Equation 4.4 and 4.5 that $cd/4 < A_0(v)/2 < A_0(v) - f_t(v) = A_t(v) < 2d$ with a probability at least $1 - o(n^{-3})$. Using $\forall 0 < x < 1/2, (1-x)^{1/x} > 1/4$ we get with a probability at least $1 - o(n^{-3})$,

$$P_t(v) = A_t(v)(1/d)(1 - 1/d)^{A_t(v)-1} \geq c/64.$$

Given this, the probability that v is not informed in $k = 128 \log n/c$ rounds is at most $(1 - c/64)^k = o(n^{-2})$.

Case 2: $p > n^{-2/5}$. In this case $T = \lfloor \log n / \log d \rfloor = 1$ and using Chernoff bounds we can show that $3d/4 < |U_2| < 3d/2$ with a probability at least $1 - o(n^{-3})$. Next we show that $A_0(v) = \Theta(dp)$ w.h.p. Since the probability that v is connected to any active node in U_2 is p, $E[A_0(v)] = |U_2|p \geq 3dp/4$ with a probability at least $1 - o(n^{-3})$. Using Chernoff bounds we get,

$$\begin{aligned} \Pr[A_0(v) < dp/2] &\leq \Pr[A_0(v) < (2/3)E[A_0(v)]] \\ &\leq e^{-E[A_0(v)](1/3)^2/2} = o(n^{-3}). \end{aligned}$$

Similarly, we get $\Pr[A_0(v) > 2dp] = o(n^{-3})$.

The rest proof is very similar to Case 1. In particular, we can show that with a probability at least $1 - o(n^{-3})$, $dp/4 < A_t(v) < 2dp$. Hence, with a probability at least $1 - o(n^{-3})$,

$$\begin{aligned} P_t(v) &= A_t(v)(1/(dp))(1 - 1/(dp))^{A_t(v)-1} \\ &> (dp/4)(1/(dp))(1 - 1/(dp))^{2dp} > 1/64. \end{aligned}$$

Thus, the probability that node v is not informed at Round k of Phase 3 is $(1 - 1/64)^k = o(n^{-2})$. Finally the lemma follows due to the Union bound. □

4.5 Gossiping in Random Networks

In this section we analyse a gossiping algorithm specialised on random networks. Furthermore, note that similar to [41, 81, 51], we can obtain a gossiping algorithm with running time $O(n \log n)$ by combining the framework proposed in [41] and the broadcasting algorithm in Section 4.4. However, the following Algorithm 8 has a better running time of $O(d \log n)$, and it uses $O(\log n)$ transmissions w.h.p. Similar to [51, 41], we assume that nodes can join messages originated from different nodes together to one large message, and we also assume that this message can be sent out in a single round. Let $m_t(u)$ be the message that is send out by node u in Round t. Then $m_1(u)$ is the message originated in u.

Algorithm 8 A gossiping algorithm for the random network $\mathbb{G}(n, p)$.

1: **for** round $r = 0$ to $128d \log n$ **do**
2: Every node transmits with probability $1/d$.
3: Every node u joins $m_r(u)$ and any incoming messages to $m_{r+1}(u)$.

Note that $d = np$ is the average node degree, and diameter

$$D = (1 + o(1))(\log n / \log d) < \log n.$$

Also, note that here nodes do not become passive after transmitting once (as it was the case in our broadcasting algorithm in Section 4.4). It is easy to see that the algorithm can be transformed into a dynamic gossiping algorithm. All that has to be done is to provide every message with a time stamp (generation time), and to delete old messages out of the $m_t(i)$ messages.

Theorem 4.5.1 *Assume $p > \delta \log n/n$ for a sufficiently large constant δ. Then, with a probability $1 - o(n^{-1})$, Algorithm 8 completes gossiping in $O(d \log n)$, and every nodes performs $O(\log n)$ transmissions w.h.p.*

Proof. First we bound the gossiping time. Let u, v $(u \neq v)$ be an arbitrary pair of nodes. Let T be the time to send the gossiping message $m_1(u)$ from u to v. Next, we show that T is w.h.p. at most $128d \log n$. Fix an arbitrary shortest path $u = u_1, \ldots u_{L+1} = v$ of length $L \leq D$ from u to v. Let T_i be the random variable representing the number of rounds that it takes node u_i to forward the first message containing $m_1(u)$ from u_i to u_{i+1}. Since u starts to submit its own message immediately in Round 1, and every node w who receives a broadcast message in Round r joins the message to its message $m_{r+1}(w)$, v will get $m_1(u)$ in Round $T \leq \sum_{i=1}^{L} T_i$. It is easy to see that the random variables $T_1, \ldots, T_L$ are independent from each other. To bound T, we first prove a result which is similar to Lemma 3.4 in [51].

Lemma 4.5.2 *Let $Y_1, \ldots, Y_L$ be a sequence of geometrically distributed random variables with parameter $1/(16d)$, i.e., $\forall 1 \leq i \leq L, k \geq 1, \Pr[Y_i = k] = 1/(16d)(1-1/(16d))^{k-1}$. Then $T \prec \sum_{i=1}^{L} Y_i$ with a probability at least $1 - o(n^{-3})$.*

Proof. The proof is similar to the proof of Lemma 3.4 in [51]. All that we have to do is to bound the probability q that a node successfully sends a message to a fixed neighbour. The expected degree of every node is d and using Chernoff bounds we can show the degree of every node is at most $2d$ with a probability $1 - o(n^{-5})$. Hence, with a probability $1 - o(n^{-5})$, we have

$$q \geq (1/d)(1 - 1/d)^{2d-1} \geq 1/(16d).$$

□

Now it remains to bound $\Pr[\sum_{i=1}^{L} Y_i \leq 128d \log n]$. Similar to the proof of Lemma 3.5 of [51], applying the standard relation of geometric distribution and binomial distributions, and using Chernoff bounds on the corresponding binomial distribution, we get

$$\begin{aligned} \Pr\left[\sum_{i=1}^{L} Y_i > 128d \log n\right] &\leq \Pr\left[\mathbb{B}(128d \log n, 1/(16d)) < L\right] \\ &\leq e^{-(7/8)^2 \cdot 8 \log n/2} = o(n^{-3}). \end{aligned}$$

The third inequality holds since $L \leq D < \log n$. The bound on the gossiping time follows by the Union bound and the fact that there are in total $n(n-1)$ source-destination pairs.

Next we bound the number of transmissions. Let v be an arbitrary node and denote Z_v to be the number of transmissions performed by v. Note that $E[Z_v] = 128 \log n$ since in each round, every node transmits with probability $1/d$ and our algorithm has in total $128d \log n$ rounds. Using Chernoff bounds we get that $Z_v \leq 256 \log n$ with probability $1 - o(n^{-2})$. By the Union bound, we get with a probability $1 - o(n^{-1})$, none of the nodes performs more than $256 \log n$ transmissions. □ □

4.6 Broadcasting in General Networks

In this section we consider broadcasting in arbitrary networks with diameter D. Czumaj and Rytter ([51]) propose a randomized algorithm with $O(\log^2 n + D \log(n/D))$ broadcasting time. Their algorithm can easily be transformed into an algorithm with the same runtime and an expected number of $\Omega(\log^2 n)$ transmissions per node. The only modification necessary is to stop nodes from transmitting after a certain number of rounds (counting onwards from the round they got the message for the first time). In Czumaj and Rytter's algorithm, each active node transmits with probability of $\Theta(1/\log(n/D))$ per round. It informs an arbitrary neighbour u (i.e. it transmits the message *and* is the only neighbour of u that transmits in that round) with a probability of $\Omega(1/(\log(n/D) \log n))$ per round. Hence, to get a high probability bound, every node has to try to send a message for $O(\log^2 n \log(n/D))$ rounds. Since an active node transmits with probability $O(1/\log(n/D))$, the total expected number of transmissions is $O(\log^2 n)$ per node. Similarly, the Algorithm 5 for unknown diameter can be transformed into an algorithm with an expected number of $O(\log^2 n)$ messages per node.

Unfortunately, in general the expected number of $O(\log^2 n)$ transmissions per node can

not be improved without increasing the broadcasting time (see Corollary 4.6.5). Under the assumption that the network diameter D is known in advance, we propose a new randomized oblivious algorithm with broadcasting time $O(D\log(n/D)+\log^2 n)$ that uses only an expected number of $O(\log^2 n/\log(n/D))$ transmissions per node (see Section 4.6.1). Note that our algorithm achieves the same broadcasting time as Algorithm 5. In Section 4.6.2, we prove a matching lower bound on the number of transmissions (Theorem 4.6.4) which indicates that our proposed algorithm is optimal in terms of the number of transmissions. In Theorem 4.6.2 we show a tradeoff between broadcasting time and number of transmissions.

4.6.1 Upper Bound for Broadcasting

In this section we show that, if the graph diameter D is known in advance, the number of transmissions can be reduced from $O(\log^2 n)$ to $O(\log^2 n/\log(n/D))$. The improvement is due to a new random distribution which is defined in Figure 4.1. Let $\lambda = \log(n/D)$. The distribution we use to generate the randomized sequence is denoted by α, and the distribution used in Section 4.1 of [51] is denoted by α'. See Figure 4.1 for a comparison of the two distributions. Note that $\forall 1 \le k \le \log n$, $1/(2\log n) \le \alpha_k \le 1/(4\lambda)$ and $\alpha_k \ge \alpha'_k/2$. Let $T = O(D\log(n/D)+\log^2 n)$ be the number of rounds for broadcasting.

$$\alpha_k = \begin{cases} \frac{1}{4\lambda} \\ \max\{\frac{1}{2\log n}, \frac{1}{2\lambda}2^{-(k-\lambda)}\} \\ \frac{1}{2\log n} \\ 1-\sum_{i=1}^{\log n}\alpha_i \end{cases} \qquad \alpha'_k = \begin{cases} \frac{1}{2\lambda} & 1 \le k \le \lambda \\ \frac{1}{2\lambda}2^{-(k-\lambda)} & \lambda < k \le \min\{\lambda+\log\log n, \log n\}, \\ \frac{1}{2\lambda\log n} & \log\log n+\lambda < k \le \log n, \\ 1-\sum_{i=1}^{\log n}\alpha'_i & \text{for } k=0. \end{cases}$$

Figure 4.1: Comparison of our distribution (left) vs. the distribution in [51] (right)

We prove the following theorem. Note that the broadcasting time is optimal according to the lower bounds shown in [76] and [81].

Theorem 4.6.1 *Algorithm 9 completes broadcasting in $O(D\log(n/D)+\log^2 n)$ rounds with probability at least $1-n^{-1}$. The expected number of messages per node is*

$$O(\log^2 n/\log(n/D)).$$

Proof. Each node is active for $O(\log^2 n)$ rounds. In every round, an active node transmits with a probability of $O(1/\log(n/D))$. Hence, the expected total number of transmissions is $O(\log^2 n/\log(n/D))$ per node.

Algorithm 9 An energy efficient broadcasting algorithm for arbitrary network with diameter D

1: Choose a randomized sequence $I =< I_1, I_2, \ldots, >$ such that $\Pr[I_r = k] = \alpha_k, \forall r \in \mathbb{N}, \forall k \in \{0, 1, \ldots, \log n\}$.
2: The status of the source is set to *active*.
3: **for** $r = 1$ to T every active node u **do**
4: Let t_u be the time step that u is informed
5: **if** $r \leq t_u + \beta \log^2 n$ (β is a constant) **then**
6: u transmits with probability 2^{-I_r}.
7: **else**
8: u becomes *passive*.
9: **if** u receives the message *for the first time* **then**
10: the status of u is set to *active*.

To show that every node receives the broadcast message, fix a round r, an arbitrary active node v and one of its neighbours w. Assume w has $m \geq 1$ active neighbours in Round r and let $1 \leq k \leq \log n$ such that $w/2 < 2^k < w$. If every active neighbour of w sends with probability 2^{-k} (i.e. $I_r = k$), w is informed with probability at least 0.1 according to Lemma 3.2 in [51]. For any $1 \leq x \leq \log n$, $\alpha_x \geq 1/(2 \log n)$, $I_r = k$ with probability at least $1/(2 \log n)$. Hence, the probability to inform w is at least $1/(20 \log n)$ per round. Using Chernoff bounds we can show that v can successfully inform all its neighbours, w.h.p.

To bound the broadcasting time, we compare the runtime of our algorithm with the runtime of the algorithm for shallow networks in [51]. Any send probability that is chosen by the algorithm in [51] is chosen with at least half the probability by our algorithm. Thus, we can use a proof that is similar to the proof of Theorem 2 in [51] to show our result. □

Finally, we demonstrate that there is a tradeoff between the expected number of transmissions and the broadcasting time.

Theorem 4.6.2 *Let* $\log(n/D) \leq \lambda \leq \log n$. *Algorithm 9 finishes broadcasting in* $O(D\lambda + \log^2 n)$ *rounds w.h.p. The expected number of transmissions is* $O(\log^2 n/\lambda)$ *per node.*

Proof. Every node is active for $O(\log^2 n)$ rounds. Moreover, the expected number of transmissions an active node performs in every round is $O(1/\lambda)$. Hence the expected total number of transmissions is $O(\log^2 n/\lambda)$ per node. Since for all $1 \leq k \leq \log n$, $\alpha_k \geq 1/(2 \log n)$, we can show (similar to the proof of Theorem 4.6.1) that every node receives the broadcasting message w.h.p.

It remains to bound the broadcasting time. Our proof is similar to the proof of Theorem 2 in [51]. We first fix some shortest path $v_0, \ldots, v_L$ of length $L \leq D$ from the source to an arbitrary node. Then, we partition all nodes into L disjoint layers with respect to that path. We assign a node u to layer $i, 1 \leq i \leq L$, if node v_i is the highest ranked node on the path that u has an edge to. In the following, a layer is called *small*, if its size is smaller than 2^λ, otherwise it is called *large*.

For an arbitrary small layer, since $\forall 1 \leq k \leq \lambda$, $\alpha_k \geq 1/(4\lambda)$, use a similar argument as in Theorem 4.6.1, we get that the probability to inform some node in the next layer is at least $1/(40\lambda)$. Hence the expected time spent on any small layer is $O(\lambda)$. Since there are at most D layers and by applying the concentration bound in Lemma 4.2.2, we get that the total time spent on all small layers is $O(D\lambda)$ w.h.p.

For an arbitrary large layer (of size $s2^\lambda, s > 1$), since $\forall \lambda < k \leq \log n$, $\alpha_k \geq \frac{1}{2\lambda} 2^{-(k-\lambda)}$, similar to Theorem 2 in [51], we can show that the probability to inform some node in the next layer is $\Omega(1/(s\lambda))$. Hence, the expected time spent on a large layer is $O(s\lambda)$. Consequently, the total expected time spent on all large layers is $O(\lambda n/2^\lambda) = O(D\lambda)$ since $2^\lambda \geq n/D$. Applying Lemma 4.2.2 once again, we obtain the high probability bound. □

4.6.2 Lower Bound on the Transmission Number

In this section we show two lower bounds for oblivious broadcasting algorithms. Observation 4.6.3, shows a lower bound on the expected number of transmissions for any randomized oblivious (every node uses the same algorithm) broadcasting algorithm. We call a probability distribution *time-invariant* if it does not depend on the time t. Theorem 4.6.4 shows a lower bound on the expected number of transmissions of any optimal randomized oblivious algorithm using a time-invariant distribution.

Observation 4.6.3 *Let A be an oblivious broadcast algorithm. Then, for every n there exists a network with $O(n)$ nodes such that A needs at least $n \log n/2$ transmissions to complete broadcasting with a probability of at least $1 - n^{-1}$.*

Proof. We construct a network with $3n + 1$ nodes. s is the node initiating the broadcast, and $d_1, \ldots, d_n$ are the destination nodes. s has an edge to $2n$ intermediate nodes $u_1, \ldots u_{2n}$. For all $1 \leq i \leq n$, d_i connects to both u_{2i-1} and u_{2i}. Let us assume that s informs $u_1, \ldots, u_{2n}$ in Round t_1. Now fix some arbitrary $T > t_1$. In Round $t_1 + 1 \leq r \leq T$, let q_r be the send probability used by the algorithm. For all $1 \leq i \leq n$, the probability to inform node d_i

in Round r is $2q_r(1-q_r)$. Due to symmetry we can assume that $q_r \leq 1/2$, resulting in $(1-q_r)^{1/q_i} \geq 1/4$. Hence,

$$\begin{aligned}
&\Pr[d_i \text{ is not informed before Round } T] \\
&= \prod_{r=t_1+1}^{T} (1-2q_r(1-q_r)) > \prod_{r=t_1+1}^{T} (1-q_r)^2 \\
&\geq \prod_{r=t_1+1}^{T} 4^{-2q_r} = 2^{-4\sum_{r=t_1+1}^{T} q_r}.
\end{aligned}$$

Now it is easy to see that, to inform d_i with probability $1-n^{-1}$, we need $\sum_{r=t_1+1}^{T} q_r > \log n/4$. Note that $\sum_{r=t_1+1}^{T} q_r$ is the expected number of transmissions that u_i and v_i perform between Round t_1+1 and T. The total number of transmissions performed by all $2n$ intermediate nodes is at least $2n\ (\log n/4) = n\log n/2$. □

Next we show a matching lower bound on the number of transmissions. This result holds for a set of randomized oblivious algorithms with optimal (i.e. $O(D\log(n/D))$) broadcasting time (e.g. the algorithm in [51]).

Theorem 4.6.4 *Let $D > 1$, let c, i be constants, and fix an arbitrary $n = 2^i$. Let A be an oblivious broadcast algorithm using a time-invariant probability distribution α. For every $n > 0$, there is a network with $O(n)$ nodes and diameter D, such that A requires an expected number of at least $\log^2 n/(\max\{4c, 8\}\log(n/D))$ transmissions per node in order to finish broadcasting in $cD\log(n/D)$ rounds with probability at least $1-n^{-1}$.*

Proof. We can assume that $D > 4\log n$, otherwise this result can be obtained directly from Observation 4.6.3 since $\log(n/D) > \log n/2$. We construct a layered network (See Figure 4.2) consisting of two subgraphs G_1 and G_2. G_1 has $\log n$ layers, namely $S_1, \ldots, S_{\log n}$, where $S_i, 1 \leq i \leq \log n$ is a star consisting of one center node c_i and 2^i leaf nodes. Every leaf node in S_i has an edge to the center c_{i+1} of S_{i+1}, for $1 \leq i \leq \log n - 1$. $G_2 = v_0, \ldots, v_L$ is a path of length $L = D - 2\log n$. To connect G_1 and G_2, we connect every node of the star $S_{\log n}$ to the first node of G_2, also denoted as $c_{\log n+1}$. Note that our network has $\sum_{i=1}^{\log n}(2^i+1) + D - 2\log n + 1 \leq 2n + D$ nodes and diameter D.

We assume that c_1 is the originator of the broadcast. The purpose of G_1 is to show that every informed node in G must be active for at least $\ln^2 n$ rounds in order to complete broadcasting with probability $1-n^{-1}$. More specifically, no matter what α is, there is always

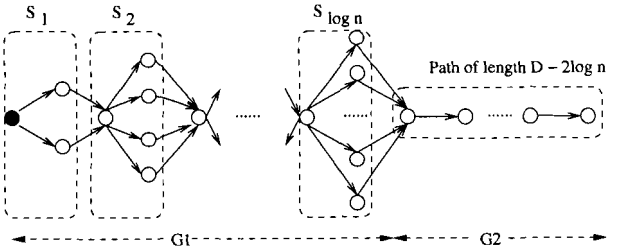

Figure 4.2: The network used in Theorem 4.6.4

a star S_i such that the probability to inform c_{i+1} is at most $1/\ln n$. Since our distribution is time invariant and every node does not know which star it belongs to, every node in the network needs to be active for at least $\ln^2 n$ rounds. Let μ be the mean of distribution α and $\Gamma(\alpha)$ be the set of outcomes of α. Next, we use G_2 to argue that in order to finish broadcasting in $cD\log(n/D)$ rounds, μ, the mean of α, must be at least $1/(2c\log(n/D))$. Hence, the total expected number of transmissions per node is at least

$$\ln^2 n(1/(2c\log(n/D)) > \log^2 n/(4c\log(n/D)).$$

Let A_i be the event that c_{i+1} is informed in Round t_i under the condition that every leaf node of S_i is active (note that they are always activated at the same time). Let Q_{t_i} be the random variable that represents the probability chosen at Round t_i. Note that Q_{t_i} has distribution α. For any $q \in \Gamma(\alpha)$, let $\Pr[A_i|Q_{t_i} = q]$ be the probability to inform c_{i+1} if $Q_{t_i} = q$. Since c_{i+1} is informed if exactly one of the 2^i leaf nodes of S_i transmits we get

$$\Pr[A_i|Q_{t_i} = q] = 2^i q(1-q)^{2^i-1} < 2^i q e^{-(2^i-1)q}. \tag{4.6}$$

Observe that $\Pr[A_i] = \sum_{q\in\Gamma(\alpha)} (\Pr[Q_{t_i} = q]\Pr[A_i|Q_{t_i} = q])$. We get,

$$\begin{aligned}
\sum_{i=1}^{\log n} \Pr[A_i] &= \sum_{i=1}^{\log n} \sum_{q\in\Gamma(\alpha)} (\Pr[Q_{t_i} = q]\Pr[A_i|Q_{t_i} = q]) \\
&= \sum_{q\in\Gamma(\alpha)} \left(\Pr[Q_{t_i} = q] \sum_{i=1}^{\log n} \Pr[A_i|Q_{t_i} = q] \right) \\
&\leq \left(\sum_{q\in\Gamma(\alpha)} \Pr[Q_{t_i} = q] \right) \frac{1}{\ln 2} = \frac{1}{\ln 2}.
\end{aligned}$$

For the third inequality, we use Equation 4.6 and

$$\forall 0 \leq q \leq 1, \int_1^\infty 2^i q e^{-(2^i-1)q} di = 1/(e^q \ln 2) \leq 1/\ln 2.$$

Consequently,

$$\min_i \Pr[A_i] \leq \left(\sum_{i=1}^{\log n} \Pr[A_i] \right) / \log n \leq \frac{1}{\ln 2 \log n} = \frac{1}{\ln n}.$$

Let $i^* = \text{argmin}_i \Pr[A_i]$. Consequently, in order to complete broadcasting with probability at least $1 - n^{-1}$, every leaf node of S_{i^*} must be active for at least $\ln^2 n$ rounds.

In the following we show $\mu \geq 1/(2c \log(n/D))$ using G_2. First note that $L = D - 2\log n > D/2$ since $D \geq 4 \log n$. For any $0 \leq i \leq L-1$, let T_i be the number of rounds that v_i is the highest ranked node on the path that is informed. Note that T_i is geometrically distributed with probability μ, we have $E[\sum_{i=0}^{L-1} T_i] = L \cdot E[T_i] = L/\mu$. Hence, in order to inform v_L within $cD \log(n/D)$ rounds (even expectedly), we need $\mu \geq 1/(2c \log(n/D))$ since $L > D/2$.

We have shown that every node in the network needs to be active for $\ln^2 n$ rounds while in each round, the expected number of transmissions it performs is at least $1/(2c \log(n/D))$. Hence, the total expected number of transmissions per node is $(\ln^2 n)(1/(2c \log(n/D))) > \log^2 n/(4c \log(n/D))$. □

Setting $D = n$ in the network constructed above, we immediately get the following corollary.

Corollary 4.6.5 *There exists a network with $O(n)$ nodes such that any randomised oblivious broadcasting algorithm that finishes broadcasting in cn rounds with probability at least $1 - n^{-1}$ requires an expected number of $\Omega(\log^2 n)$ transmissions.*

4.7 Summary

We have considered an "energy efficient" routing model for ad hoc networks. Our goal is to minimize not only the broadcasting and gossiping time, but also the the energy consumption, which is measured by the total number of messages sent. For random networks, we presented a $O(\log n)$ broadcasting algorithm where every node transmits at most once and a $O(d \log n)$ gossiping algorithm using $O(\log n)$ messages per node. For general networks with known diameter D, we presented a randomized broadcasting algorithm with optimal broadcasting time $O(D \log(n/D) + \log^2 n)$ that uses $O(\log^2 n / \log(n/D))$ transmissions per

node in expectation. Our lower bound $\Omega(\log^2 n/\log(n/D))$ on the number of transmissions matches our upper bound for time-invariant distributions. We also demonstrated a tradeoff between these two objectives.

There are a few interesting directions for future work. First, so far we used the Erdös-Rényi model to model practical ad hoc networks, which is somewhat unrealistic. We can consider other alternative models for random graphs, such as the random geometric graphs [95]. Second, the question remains open to determine the minimum energy consumption for gossiping in general networks. Third, it would be interesting to generalize the lower bound result in Theorem 4.6.4 for general distributions without the time-invariant property. Last but not least, similar to [39], we can consider the more general problem that $1 \leq k \leq n$ different nodes initiate broadcasting. Note that broadcasting (i.e., $k = 1$) and gossiping (i.e., $k = n$) are two special cases of this problem.

Chapter 5

Conclusion

In this thesis, we have studied distributed algorithms for two fundamental problems in distributed systems, resource allocation and routing. We considered two well-motivated resource allocation models, the diffusive load balancing and the weighted balls-into-bins games. We also studied routing algorithms for broadcasting and gossiping on ad hoc networks.

Diffusive load balancing Diffusive load balancing, a typical neighbourhood load balancing model, studies how nodes with some initial tasks in a network balance their loads concurrently with all their neighbours. The concurrent load exchanging actions have been the main obstacle for the analysis of the diffusion algorithms since the load situation could change significantly during one single step of load exchanges. In this thesis, we have proposed a novel analytical method. The idea is to first sequentialize concurrent actions to obtain a sequential system, analyze the sequential system, and then bound the gap between both systems. We have demonstrated the strength of this technique by analyzing several diffusion algorithms. This idea is simple yet also general. We believe that it is helpful in the analysis of many other distributed systems with concurrent actions. In particular, we have applied the same idea to analyze the selfish-allocation game in Chapter 3.

Weighted balls-into-bins games The weighted balls-into-bins game studies how to allocate a set of weighted balls into a set of bins in a balanced manner. We have considered two different scenarios, the static sequential game and the selfish reallocation game.

In the static sequential game, balls comes one after another and have to be allocated in such order. We have considered a well-known approach that to have every ball pick

$d \geq 1$ bins independently and uniformly at random and place itself into the least loaded bin. We have studied how the outcome of the game, the expected maximum load of any bin, is influenced by the game parameters such as the distribution of ball weights, and the order that balls are allocated. Our main idea is to use the majorization technique inductively to show one system majorizes another. In particular, we have shown that the single-choice game ($d = 1$) is "order-preserving" according to vector majorization while the multiple-choice game ($d \geq 2$) does not have this nice property. We have also discussed several limitations of this technique. For future work, we can apply this technique to study other related problems in the static sequential game. For example, given two systems with the same total weight, does the system with small number of "large" balls always majorizes the system with large number of "tiny" balls? Or whether their corresponding expected maximum loads differ only by a constant? However, we believe that new ideas are necessary to answer this question.

In the selfish reallocation game, every ball has its own initial location. We have studied an iterative, selfish distributed reallocation algorithm. We have shown some upper and lower bounds for the convergence time of the algorithm, which is the number of steps for the system to terminate upon reaching (or getting close to) the Nash equilibrium. Our main proof method is the potential function technique. The idea is to define some potential function to measure the distance between some system state and the Nash equilibrium, and then to show that the potential always decreases in expectation. For the uniform case where each ball is of uniform weight, we obtained tight bounds for the convergence time. To our best knowledge, this work is the first attempt to analyze the selfish reallocation game with heterogenous tasks. In the future, we can consider applying our proof technique to study more general models, for example, those ones that allow arbitrary latency functions.

Energy Efficient Routing in Ad Hoc Networks We have considered an "energy efficient" ad hoc network model, in which the energy consumption of a broadcasting/gossiping algorithm is measured in terms of the total number of messages (or transmissions) sent. Our goal is two-fold: we want to minimize not only broadcasting/gossiping time, but also energy consumption. Under this model, we have presented and analyzed several energy efficient broadcasting/gossiping algorithms for both random and general ad hoc networks. We have also given some lower bounds for the energy consumption, and demonstrated a tradeoff between these two objectives. In the future, we can try to generalize this model

to more practical scenarios. For instance, we can assume that our ad hoc network has a topology of random geometric graph similar to [95].

In this thesis we have studied distributed algorithms for two fundamental problems in distributed systems, resource and routing. Although the models we consider are different, they do share common features and challenges, such as lack of central control (knowledge) and concurrent user actions. We have presented various new results as well as some novel proof techniques. IIn the future, we plan to further study the scope and limitation of the proposed techniques. We also plan to exploit the practical implications of our theoretical results by studying the corresponding real-life problems.

Chapter 6

Appendix

6.1 Tail Estimates

The following version of Chernoff bounds can be found, for example, in, [91].

Lemma 6.1.1 *Let $X_1, \ldots X_n$ be independent Bernoulli random variables and let $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then we have,*

1. $\Pr[X < (1-\epsilon)\mu] < e^{-\mu\epsilon^2/2}$, *for* $0 \leq \epsilon \leq 1$.
2. $\Pr[X > (1+\epsilon)\mu] < e^{-\mu\epsilon^2/3}$, *for* $\epsilon > 0$.
3. $\Pr[|X-\mu| \leq \epsilon\mu] > 1 - 2e^{-\mu\epsilon^2/3}$, *for* $0 \leq \epsilon \leq 1$.

6.2 An Alternative Proof of Theorem 3.4.8

After we submitted [24] to a journal, one of the anonymous reviewers pointed out the proof of Lemma 3.4.4 can in fact be simplified. The following proof is rewritten based on the comments.

Fix an arbitrary allocation $\omega \in \Omega^n$. We first prove a lemma which indicates that Function $f(w) = S_i(w, \omega)$ is convex.

Lemma 6.2.1 *Function $f(w) = S_i(w, \omega)$ is convex.*

Proof. Let v and v' be two m-dimensional vectors such that $w = (1-\lambda)v + \lambda v'$.

$$S_i(w,\omega) = \max_{A \subset [n], |A| = i} \sum_{1 \leq j \leq m : \omega_j \in A} w_j$$

$$
\begin{aligned}
&= \max_{A\subset[n],|A|=i} \sum_{1\le j\le m:\omega_j\in A} ((1-\lambda)v_j + \lambda v'_j) \\
&\le (1-\lambda) \max_{B\subset[n],|B|=i} \sum_{1\le j\le m:\omega_j\in B} v_j + \\
&\quad \lambda \max_{B'\in[n],|B'|=i} \max_{B'\subset[n],|B'|=i} \sum_{1\le j\le m:\omega_j\in B'} v'_j \\
&= (1-\lambda)S_i(v,w) + \lambda S_i(v',w).
\end{aligned}
$$

□

Note that if $w \xrightarrow{T} w'$, $w' = \lambda \cdot w + (1-\lambda)wP$, where P is a permutation matrix. Using Lemma 6.2.1, we get

$$
\begin{aligned}
E[S_i(w')] &\le \lambda E[S_i(w)] + (1-\lambda)E[S_i(wP)] \\
&= E[S_i(w)].
\end{aligned}
$$

The second equation holds since $E[S_i(w)] = E[S_i(wP)]$ for any permutation matrix P.

Bibliography

[1] Micah Adler, Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul Goldberg, and Mike Paterson. A proportionate fair scheduling rule with good worst-case performance. In *Proc. 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 101–108, 2003.

[2] Micah Adler, Petra Berenbrink, and K. Schröder. Analyzing an infinite parallel job allocation process. In *Proc 6th European Symposium on Algorithms (ESA)*, pages 417–428, 1998.

[3] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. Parallel randomized load balancing. In *Proc 27th ACM Symposium on Theory of computing (STOC)*, pages 238–247, 1995.

[4] William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *Proc 32nd ACM Symposium on Theory of computing (STOC)*, pages 171–180, 2000.

[5] David J. Aldous and James A. Fill. *Reversible Markov Chains and Random Walks on Graphs.* http://www.stat.berkeley.edu/ aldous/RWG/book.html. Book in preparation.

[6] Romas Aleliunas. Randomized parallel communication. In *Proc 1stACM Symposium on Principles of Distributed Computing (PODC)*, pages 60–72, 1982.

[7] Noga Alon, Amotz Bar-Noy, Nathan Linial, and David Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.

[8] Matthew Andrews, Antonio Fernández, Ashish Goel, and Lisa Zhang. Source routing and scheduling in packet networks. *Journal of ACM*, 52(4):582–601, 2005.

[9] Elliot Anshelevich, David Kempe, and Jon Kleinberg. Stability of load balancing algorithms in dynamic adversarial systems. In *Proc 34th ACM Symposium on Theory of computing (STOC)*, pages 399–406, 2002.

[10] Hagit Attiya and Jennifer Welch. *Distributed Computing.* Wiley and Sons Inc., 2004.

[11] Friedhelm Meyer auf der Heide, Christian Scheideler, and Volker Stemann. Exploiting storage redundancy to speed up randomized shared memory simulations. In *Proc 12nd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 267–278, 1995.

[12] Baruch Awerbuch, Petra Berenbrink, Andre Brinkmann, and Christian Scheideler. Simple routing strategies for adversarial systems. In *Proc 42th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 158–167, 2001.

[13] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal on Computing*, pages 180–200, 1999.

[14] Reuven Bar-Yehuda, Oded Goldreich, and Alon Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

[15] Reuven Bar-Yehuda, Amos Israeli, and Alon Itai. Multiple communication in multihop radio networks. *SIAM Journal Computing*, 22(4):875–887, 1993.

[16] Petra Berenbrink. *Randomized Allocation of Independent Tasks*. PhD thesis, Department of Computer Science, University of Pardeborn, 2000.

[17] Petra Berenbrink, Friedhelm Meyer auf der Heide, and Klaus Schröder. Allocating weighted balls in parallel. *Theory of Computing Systems*, 32:281–300, 1999.

[18] Petra Berenbrink, Colin Cooper, and Zengjian Hu. Energy efficient randomised communication in unknown adhoc networks. In *Proc. 19th ACM Symposium on Parallelism in Algorithms and Architectures(SPAA)*, 2007.

[19] Petra Berenbrink, Artur Czumaj, Tom Friedetzky, and Nikita Vvedenskaya. Infinite parallel job allocation (extended abstract). In *Proc. 14th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 99–108, 2000.

[20] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: the heavily loaded case. In *Proc 32th ACM Symposium on Theory of Computing (STOC)*, pages 745–754, 2000.

[21] Petra Berenbrink, Tom Friedetzky, and Leslie A. Goldberg. The natural work-stealing algorithm is stable. In *Proc 42th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 178–187, 2001.

[22] Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul W. Goldberg, Zengjian Hu, and Russell A. Martin. Distributed selfish load balancing. In *Proc 17th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 354–363, 2006.

[23] Petra Berenbrink, Tom Friedetzky, and Zengjian Hu. A new analytical method to parallel, diffusion-type load balancing. In *Proc 21st IEEE International Parallel and Distributed Processing Symposium(IPDPS)*, page 56, 2006.

[24] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Iman Hajirasouliha. Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks, 2007.

[25] Petra Berenbrink, Tom Friedetzky, Zengjian Hu, and Russell Martin. On weighted balls-into-bins games. In *Proc 22nd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 231–243, 2005.

[26] Petra Berenbrink, Tom Friedetzky, and Russell Martin. Dynamic diffusion load balancing. Technical report, University of Warwick, 2004.

[27] Petra Berenbrink, Tom Friedetzky, and Ernst W. Mayr. Parallel continuous randomized load balancing. In *Proc. 11th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 192–201, 1998.

[28] Petra Berenbrink, Tom Friedetzky, and Angelika Steger. Randomized and adversarial load balancing. In *Proc. 11th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 175–184, 1999.

[29] Petra Berenbrink, Marco Riedel, and Christian Scheideler. Simple competitive request scheduling strategies. In *Proc. 11th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 33–42. Association for Computing Machinery, 1999.

[30] Petra Berenbrink and Christian Scheideler. Locally efficient on-line strategies for routing packets along fixed paths. In *Proc 10th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 112–121, 1999.

[31] Jacques E. Boillat. Load balancing and poisson equation in a graph. *Concurrency - Practice and Experience*, 4:289–314, 1990.

[32] Belá Bollobás. The diameter of random graphs. *IEEE Transactions on Information Theory*, 36(2):285–288, 1990.

[33] Russ Bubley and Martin E. Dyer. Path coupling: A technique for proving rapid mixing in markov chains. In *Proc 38th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 223–231, 1997.

[34] Julien Cartigny, David Simplot, and Ivan Stojmenović. Localized minimum-energy broadcasting in ad-hoc networks. In *Proc 22nd Annual IEEE Conference on Computer Communications*, pages 2210–2217, 2003.

[35] Lijun Chen, Steven H. Low, Mung Chiang, and John C. Doyle. Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks. In *Proc 25th Annual IEEE Conference on Computer Communications*, 2006.

[36] Steve Chien and Alistair Sinclair. Convergence to approximate nash equilibria in congestion games. In *Proc 18th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 2007.

[37] Bogdan S. Chlebus, Leszek Gasieniec, Alan Gibbons, Andrzej Pelc, and Wojciech Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing*, 15(1):27–38, 2002.

[38] Bogdan S. Chlebus, Leszek Gasieniec, Anna Östlin, and John Michael Robson. Deterministic radio broadcasting. In *Proc 27th International Colloqium on Automata, Languages and Programming (ICALP)*, pages 717–728, 2000.

[39] Bogdan S. Chlebus, Leszek Gasieniec, and Wojciech Rytter. Fast broadcasting and gossiping on radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.

[40] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. Average-time complexity of gossiping in radio networks. In *Proc 13th Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 253–267, 2006.

[41] Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. Fast broadcasting and gossiping in radio networks. In *Proc 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–581, 2000.

[42] Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. A randomized algorithm for gossiping in radio networks. In *Proc 7th International Computing and Combinatorics Conference (COCOON)*, pages 483–492, 2001.

[43] Fan Chung and Linyuan Lu. The diameter of random sparse graphs. *Advances in Applied Mathematics*, 26(4):257–279, 2001.

[44] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[45] Andrea E. F. Clementi, Angelo Monti, and Riccardo Silvestri. Distributed broadcasting in radio networks with unknown topology. *Theoretcal Computing Science*, 1-3(302):337–364, 2003.

[46] Rene L. Cruz and Arvind Santhanam. Optimal routing, link scheduling, and power control in multi-hop wireless networks. In *Proc 22nd Annual IEEE Conference on Computer Communications*, pages 702–711, 2003.

[47] George Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel Distributed Computer Systems*, 7(2):279–301, 1989.

[48] A. Czumaj, C. Riley, and C. Scheideler. Perfectly balanced allocation. In *Proc 11th International Workshop on Randomization and Approximation Techniques in Computer Science(RANDOM)*, pages 240–251. LNCS, 2003.

[49] Artur Czumaj. Recovery time of dynamic allocation processes. *Theory of Computer Systems*, 33(5/6):465–487, 2000.

[50] Artur Czumaj. Selfish routing on the internet. In Joseph Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC press, 2003.

[51] Artur Czumaj, Piotr Krysta, and Berthold Vöcking. Selfish traffic allocation for server farms. In ACM, editor, *Proc 34th ACM Symposium on Theory of Computing (STOC)*, pages 287–296, 2002.

[52] Artur Czumaj and Wojciech Rytter. Broadcasting algorithms in radio networks with unknown topology. *Journal Algorithms*, 60(2):115–143, 2006.

[53] Artur Czumaj and Volker Stemann. Randomized allocation processes (extended abstract). In *Proc 38th Symposium on Foundations of Computer Science (FOCS)*, pages 194–203, 1997.

[54] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. In *Proc 13th ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 413–420, 2002.

[55] Ralf Dickmann, Andreas Frommer, and Burkhard Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25:789–812, 1999.

[56] R. Elsässer and L. Gasieniec. Radio communication on random graphs. In *Proc. 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 309–315, 2005.

[57] Robert Elsässer. On the communication complexity of randomized broadcasting in random-like graphs. In *Proc. 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 148–157, 2006.

[58] Robert Elsässer and Burkhard Monien. Load balancing of unit size tokens and expansion properties of graphs. In *Proc. 15th ACM Symposium on Parallelism in Algorithms and Architectures(SPAA)*, pages 266–273, 2003.

[59] Robert Elsässer, Burkhard Monien, and Stefan Schamberger. Load balancing in dynamic networks. In *Proc 7th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 193–200, 2004.

[60] Eyal Even-Dar, Alexander Kesselman, and Yishay Mansour. Convergence time to nash equilibria. In *Proc 30th International Colloqium on Automata, Languages and Programming (ICALP)*, pages 502–513, 2003.

[61] Eyal Even-Dar and Yishay Mansour. Fast convergence of selfish rerouting. In *Proc 16th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 772–781, 2005.

[62] Simon Fischer, Harald Räcke, and Berthold Vöcking. Fast convergence to wardrop equilibria by adaptive sampling methods. In *Proc 38th ACM Symposium on Theory of computing (STOC)*, pages 653–662, 2006.

[63] Bhaskar Ghosh, Frank Thomson Leighton, Bruce M. Maggs, S. Muthukrishnan, C. Greg Plaxton, Rajmohan Rajaramanand Andra W. Richa, Robert Endre Tarjan, and David Zuckerman. Tight analyses of two local load balancing algorithms. In *Proc 27th ACM symposium on Theory of computing (STOC)*, pages 548–558, 1995.

[64] Bhaskar Ghosh and S. Muthukrishnan. Dynamic load balancing in parallel and distributed networks by random matchings (extended abstract). In *Proc. 6th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 226–235, 1994.

[65] Paul W. Goldberg. Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In *Proc 23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 131–140, 2004.

[66] Rajiv Gupta, Scott A. Smolka, and Shaji Bhaskar. On randomization in sequential and distributed algorithms. *ACM Computer Survey*, 26(1):7–86, 1994.

[67] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proc 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking(MOBICOM)*, pages 174–185, 1999.

[68] Mark Jerrum. *Counting, sampling and integrating: algorithms and complexity.* Springer Verlag, 2003.

[69] Richard M. Karp, Michael Luby, and Friedhelm Meyer auf der Heide. Efficient pram simulation on a distributed memory machine. *Algorithmica*, 16(4/5):517–542, 1996.

[70] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Andrzej Pelc. Power consumption in packet radio networks (extended abstract). In *Proc. 14th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 363–374, 1997.

[71] Elias Koutsoupias, Marios Mavronicolas, and Paul G. Spirakis. Approximate equilibria and ball fusion. *Theory of Computer Systems*, 36:683–693, 2003.

[72] Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Proc 16th Symposium on Theoretical Aspects of Computer Science (STACS)*, Lecture Notes in Computer Science, pages 404–413. Springer, 1999.

[73] Dariusz R. Kowalski and Andrzej Pelc. Deterministic broadcasting time in radio networks of unknown topology. In *Proc 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 63–72, 2002.

[74] Dariusz R. Kowalski and Andrzej Pelc. Broadcasting in undirected ad hoc radio networks. In *Proc 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 73–82, 2003.

[75] Dariusz R. Kowalski and Andrzej Pelc. Faster deterministic broadcasting in ad hoc radio networks. In *Proc 20th Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 109–120, 2003.

[76] T.G. Kurtz. Strong approximation theorems for density dependent markov chains. *Stochastic Processes and Applications*, 6:223–240, 1978.

[77] Eyal Kushilevitz and Yishay Mansour. An omega(log (/d)) lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

[78] F. T. Leighton, Bruce M. Maggs, Abhiram G. Ranade, and Satish B. Rao. Randomized routing and sorting on fixed-connection networks. *Journal of Algorithms*, 17(1):157–205, 1994.

[79] Frank Thomson Leighton, Bruce M. Maggs, and Satish Rao. Packet routing and job-shop scheduling in *o*(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[80] Frank Thomson Leighton, Bruce M. Maggs, and Andréa W. Richa. Fast algorithms for finding o(congestion + dilation) packet routing schedules. *Combinatorica*, 19(3):375–401, 1999.

[81] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *Proc 4th ACM Conference on Electronic Commerce (EE)*, pages 36–41, 2003.

[82] Ding Liu and Manoj Prabhakaran. On randomized broadcasting and gossiping in radio networks. In *Proc 8th International Computing and Combinatorics Conference (COCOON)*, pages 340–349, 2002.

[83] Linyuan Lu. The diameter of random massive graphs. In *Proc 12th annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 912–921, 2001.

[84] Michael Luby, Dana Randall, and Alistair R. Sinclair. Markov chain algorithms for planar lattice structures. *SIAM Journal on Computing*, 31:167–192, 2001.

[85] Nancy Lynch. *Distributed Algorithms.* Morgan Kaufman Publishers Inc., 1996.

[86] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *Proc 4th Symposium on Operating Systems Design and Implementation (OSDI)*, pages 131–146. 2002.

[87] Albert M. Marshall and Ingram Olkin. *Inequalities, Theory of Majorization and its Applications.* Academic Press, 1979.

[88] Michael Mitzenmacher. Load balancing and density dependent jump markov processes. In *Proc 37st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 213–222, 1996.

[89] Michael Mitzenmacher. The power of two choices in randomized load balancing. *Journal of Parallel and Distributed Systems*, 12:1094–1104, 2001.

[90] Michael Mitzenmacher, Balaji Prabhakar, and Devavrat Shah. Load balancing with memory. In *Proc 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 799–808, 2002.

[91] Michael Mitzenmacher, Andrea W. Richa, and Ramesh Sitaraman. The power of two random choices: A survey of technique and results. *Handbook of Randomized Computing*, 2000.

[92] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge Press, 2005.

[93] Prasant Mohapatra. Wormhole routing techniques for directly connected multicomputer systems. *ACM Computer Survey*, 30(3):374–410, 1998.

[94] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge Press, 1995.

[95] S. Muthukrishnan, Bhaskar Ghosh, and Martin H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computer Systems*, 4:331–354, 1998.

[96] S. Muthukrishnan and Gopal Pandurangan. The bin-covering technique for thresholding random geometric graph properties. In *Proc. 16th ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 989–998, 2005.

[97] Christos Papadimitriou. Algorithms, games, and the internet. In *Proc 33rd ACM Symposium on Theory of computing (STOC)*, pages 749–753, 2001.

[98] David Peleg and Eli Upfal. The token distribution problem. *SIAM journal on Computing*, 2:229–243, 1989.

[99] Yuval Rabani, Alistair Sinclair, and Rolf Wanka. Local divergence of markov chains and the analysis of iterative load balancing schemes. In *Proc 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 694–705, 1998.

[100] Rajmohan Rajaraman. Topology control and routing in ad hoc networks: a survey. *SIGACT News*, 33(2):60–73, 2002.

[101] Tim Roughgarden. How unfair is optimal routing? In *Proc 13th ACM-SIAM symposium on Discrete Algorithms (SODA)*, pages 203–204, 2002.

[102] Tim Roughgarden. The price of anarchy is independent of the network topology. In *Proc 34th ACM Symposium on Theory of computing (STOC)*, pages 428–437, 2002.

[103] Tim Roughgarden. *Selfish routing.* PhD thesis, Department of Computer Science, Cornell University, 2002.

[104] Tim Roughgarden and Éva Tardos. How bad is selfish routing? *Journal of ACM*, 49(2):236–259, 2002.

[105] Peter Sanders. On the competitive analysis of randomized static load balancing. In *Proc. 1st Workshop on Randomized Parallel Algorithms (RANDOM)*, 1996.

[106] Peter Sanders, Sebastian Egner, and Jan Korst. Fast concurrent access to parallel disks. *Algorithmica*, 35(1):21–55, 2003.

[107] Volker Stemann. Parallel balanced allocations. In *Proc. 8th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 261–269, 1996.

[108] Raghu Subramanian and Isaac D. Scherson. An analysis of diffusive load-balancing. In *Proc. 6th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 220–225, 1994.

[109] Eli Upfal. Efficient schemes for parallel communication. *Journal of ACM*, 31(3):507-517, 1984.

[110] Adrian Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *Proc 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 416–428, 2002.

[111] Berthold Vöcking. How asymmetry helps load balancing. *Journal of ACM*, 50(4):568-589, 2003.

[112] Cheng-Zhong Xu, Burkhard Monien, Reinhard Lüling, and Francis C. M. Lau. An analytical comparison of nearest neighbor algorithms for load balancing in parallel computers. In *Proc 9th International Parallel Processing Symposium (IPPS)*, pages 472–479, 1995.

[113] Ying Xu. An $o(n^{1.5})$ deterministic gossiping algorithm for radio networks. *Algorithmica*, 36(1):93–96, 2003.