DISCRETE DISTRIBUTED LOAD BALANCING

by

Hoda Akbari

M.Sc., Sharif University of Technology, 2009

B.Sc., Sharif University of Technology, 2007

A Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2014

APPROVAL

Name:	Hoda Akbari
Degree:	Doctor of Philosophy
Title:	Discrete Distributed Load Balancing
Examining Committee:	**Chair**: Dr. Binay Bhattacharya Professor
	Dr. Petra Berenbrink, Associate Professor, Computing Science Simon Fraser University Senior Supervisor
	Dr. Funda Ergun, Professor, Computing Science Simon Fraser University Supervisor
	Dr. Andrei Bulatov, Professor, Computing Science Simon Fraser University SFU Examiner
	Dr. Nicholas J. A. Harvey, Assistant Professor Department of Computer Science University of British Columbia External Examiner
Date Approved:	Nov. 20, 2014

Partial Copyright Licence

Abstract

The neighbourhood load balancing problem considers a network along with a distribution of tasks over its nodes. The aim is to minimize *discrepancy* which is the difference between the maximum and the minimum load. This is done by a distributed process that exchanges load between neighbouring nodes. One distinguishes between the continuous model - where tasks can be split arbitrarily - and the more practical discrete model in which tasks are atomic. The former has been comprehensively studied in early results [20, 57, 29]. In the latter model, various algorithms have been developed and analyzed [10, 9, 27, 28, 41, 39, 44, 57, 62, 64]. Nevertheless, several aspects are yet to be explored, such as devising new discrete algorithms, and generalizing or tightening the existing analyses.

We study the problem of discrete neighbourhood load balancing. We propose a new approach that can transform continuous load balancing algorithms into deterministic or randomized discrete versions. Despite its simplicity, the proposed approach works in quite general settings (arbitrary network topology, weighted tasks and heterogeneous processors) and in many cases achieves improved discrepancy bounds. We also study the usage of *rotor-router walks* – deterministic analogue of random walks – for discrete load balancing, and in particular, derandomization of existing randomized approaches. After that, we obtain discrepancy bounds for deterministic and randomized discrete *second-order* processes [57], where the amount of load transferred in each round depends both on the current load distribution and the amount of load transferred in the previous round. In second-order processes a node may attempt to send out more load than its available load. To address this issue, we provide bounds on the minimum load of any node which is sufficient to prevent such conditions.

Acknowledgments

Completion of this thesis was possible with the help and support of several people. Foremost, I would like to express my sincere gratitude to my supervisor Dr. Petra Berenbrink for the continuous support of my PhD study and research, for her patience, motivation, insight, and helpful discussions. I would like to thank the rest of my thesis committee: Dr. Funda Ergun, Dr. Andrei Bulatov, and Dr. Nicholas Harvey for their time and generous support to improve the quality of this work. Special thanks to Dr. Thomas Sauerwald and Dr. Robert Elsässer with whom I had the honour of research collaboration and for their very helpful discussions. I am also thankful to Dr. Tom Friedetzky for his guidance on how to better present the material.

I am also indebted to my fellow labmates for stimulating discussions. Thanks also go to all faculty, staff, and friends in the School of Computing Science at SFU for providing such a nice academic environment. My graduate studies would not have been interesting without them. I also thank Simon Fraser University, for scholarship funding that helped me to focus full time on my research.

I cannot thank my parents enough. This journey would not have been possible without their encouragement and endless support. Finally, I extend heartfelt thanks to my husband, for his continued love and staying with me throughout this journey.

Contents

Chapter 1

Introduction

Due to the ubiquity of computer networks in a wide variety of applications, distributed computation on large decentralized networks is becoming an increasingly popular area of research. Here, load balancing is a key issue with considerable impact on the performance of the systems and the perceived quality of service. The goal of load balancing is that the processors receive equal shares of the total workload or data. Nevertheless, an overall benefit is achieved only if the balancing process itself is very efficient. The work proposed here centres on design and analysis of efficient balancing processes.

The neighbourhood load balancing problem considers a network along with a distribution of tasks over its nodes. The aim is to minimize *discrepancy* which is the difference between the maximum and the minimum load. This is done by a distributed process that exchanges load between neighbouring nodes. Chapter 2 contains an overview of the existing results in this area. Early results considered the *continuous* model where tasks can be split arbitrarily. In the more realistic *discrete* model, on the other hand, tasks are atomic and cannot be divided to smaller tasks. As will be mentioned in Chapter 2, the continuous model is well understood (Section 2.1), while the discrete model is still attracting researchers' attention. Various discrete algorithms have been developed and analyzed (Section 2.2). Nevertheless, several aspects are yet to be explored, such as devising new discrete algorithms, and improving the existing analyses.

We study the problem of discrete neighbourhood load balancing. In Chapter 3, we present a new approach that transforms continuous load balancing algorithms into deterministic or randomized discrete versions. Chapter 4 contains new results on the usage of *rotor-router walks* – deterministic analogue of random walks – for discrete load

balancing, and in particular, derandomization of existing randomized approaches. In Chapter 5 we analyze discrete *second-order* processes [57] (processes where the amount of load transferred in each round depends both on the current load distribution and the amount of load transferred in the previous round).

Next, we describe the model more formally and introduce the notation.

1.1 Model and Notation

In the neighbourhood load balancing problem we consider a connected network of n processors. The network is modelled by a graph $G = (V, E)$ of order n with the processors as its nodes labelled by $V = \{1, \ldots, n\}$ and the communication links as its edges. $N(i)$ is the *neighbourhood* of node i, defined as the set of direct neighbours of i. The degree of a node i is $d_i = |N(i)|$, and d is the maximum degree among all the nodes. Initially, m tasks are arbitrarily distributed over the nodes.

In general settings, both nodes and tasks have statically assigned positive weights. The weight of a node i is denoted by s_i which models the amount of computational resources i has, or simply its speed. We assume $s_1 = 1$, and for all $2 \leqslant i \leqslant n$, we have $s_i \geqslant 1$. We define $s = s_1 + \cdots + s_n$ as the *capacity* of the network. The network is called *homogeneous* when all the nodes have the same speed and *heterogeneous* otherwise. Similarly, weight w_i of a task i is proportional to the amount of computational resources i requires in order to finish. When tasks are identical, they are also called *tokens*. The maximum task weight is denoted by $w_{\max}$, and $w := w_1 + \cdots + w_m$ is the total weight of all the tasks.

The *load* x_i of a node i is defined as the total weight of its tasks. The *makespan* ℓ_i of node i is defined as its load divided by its speed, i.e., x_i/s_i. The makespan of an assignment $(x_1, \ldots, x_n)$ is the maximum makespan of any node. The goal is to distribute the tasks so that the makespans of all the nodes are equal and thus the makespan of the assignment is minimized. In other words, the balancing process should eventually converge to a state where $x_i = \bar{x}_i := s_i \cdot w/s$. In a homogeneous network, since all $\bar{x}_i$ are equal we drop the node index and denote the average load by $\bar{x} := m/n$. We consider synchronous processes, i.e., processes that work in synchronous rounds. Each round consists of each node exchanging load with its neighbours in parallel in a decentralized fashion. The key issue is how to determine the amount of load to be sent to each of the neighbours. A general scheme is shown in Algorithm 1.1, which is run in parallel over

all the nodes.

For a process $\mathcal{P}$, the load vector at the beginning of round t is denoted by

$$x^{\mathcal{P}}(t) = (x_1^{\mathcal{P}}(t), \ldots, x_n^{\mathcal{P}}(t))$$

Also, $y_{i,j}^{\mathcal{P}}(t)$ represents the load transferred from node i to node j, so that we have:

$$x_i^{\mathcal{P}}(t+1) = x_i^{\mathcal{P}}(t) - \sum_{j \in N(i)} (y_{i,j}^{\mathcal{P}}(t) - y_{j,i}^{\mathcal{P}}(t)). \tag{1.1}$$

When clear from the context, we omit the process superscript.

Algorithm 1.1 A General Neighbourhood Load Balancing Algorithm: The process on node i at round t

for each neighbour j of i in parallel
Compute $y_{i,j}^{\mathcal{P}}(t)$
Send a total load of $y_{i,j}^{\mathcal{P}}(t)$ to j
$y_{j,i}^{\mathcal{P}}(t) \leftarrow$ The amount of load received from j

$x_i^{\mathcal{P}}(t+1) = x_i^{\mathcal{P}}(t) - \sum_{j \in N(i)} (y_{i,j}^{\mathcal{P}}(t) - y_{j,i}^{\mathcal{P}}(t))$

The *discrepancy*[1] of a vector v is defined as $(\max_i v_i - \min_i v_i)$. The widely accepted balancing measure is the *makespan discrepancy* which is the discrepancy of the makespan vector. In a homogeneous network this is equivalent to the *load discrepancy*, which is the discrepancy of the load vector. Here, we will use the shortened term *discrepancy* for the makespan discrepancy. We also define *max-avg discrepancy* as the difference between the maximum makespan and w/s, which is the makespan of the balanced allocation.

The *convergence time* or the *balancing time* of a continuous load balancing process $\mathcal{P}$ – denoted by $T^{\mathcal{P}}$– is the time it takes for the algorithm until it reaches a state with constant discrepancy – e.g. discrepancy of one. In this thesis, we use the following definition which ensures a discrepancy of two:

$$T^{\mathcal{P}} := T^{\mathcal{P}}(x(0)) = \{\min t : \forall i, |x_i(t) - w \cdot s_i/s| \leqslant 1\}.$$

When clear from the context, we omit the process superscript of T. In the context of randomized processes, we say an event occurs *with high probability (w.h.p)* if its probability is at least $1 - \mathcal{O}(n^{-\alpha})$ for some constant $\alpha > 0$.

[1]For more clarity, we will sometimes use the term "max-min discrepancy".

Tables 1.1 and 1.2 outline the notation and terms used throughout the thesis.

1.2 An Overview of the Results

This thesis extends and contributes to the existing results in the discrete distributed load balancing from different angles. In Chapter 3, we present a new approach through which a discrete process tries to achieve load distribution similar to a continuous process. In Chapter 4 we analyze a so-called rotor walk model as a model for load balancing, leading to new results in both the rotor walks and the load balancing area. Chapter 5 extends the existing results on randomized discrete load balancing to heterogeneous networks and so-called second-order processes by generalizing the existing analysis techniques. This was made possible by introducing abstractions on some concepts such as *rounding*.

The main idea of Chapter 3 is to consider cumulative values of load transferred over each edge, while previous algorithms consider the current state of the network only. This provides a framework that can transform continuous algorithms to discrete algorithms in quite general settings (heterogeneous networks, weighted tasks, and variety of continuous algorithms). A deterministic and a randomized transformation are presented (task weights are not supported in the randomized algorithm). Both algorithms calculate the cumulative values in the same way. The former algorithm rounds the values down while the latter applies randomized rounding. Let T be the convergence time of the continuous process. Then at time T the deterministic and randomized transformation achieve max-avg discrepancies of $\mathcal{O}(d \cdot w_{\max})$ (Theorem 3.3) and $\frac{d}{4} + \mathcal{O}(\sqrt{d \log n})$ (Theorem 3.8), respectively.

In Chapter 4 we consider the *rotor walk* (or Propp machine) model which distributes tokens in a round-robin fashion. The model was originally introduced as a derandomized version of a random walk model in which tokens perform random walk steps independently in a synchronous manner. To measure how well rotor walks can approximate random walks one compares the number of tokens a node has in the two models, taking the maximum difference between the two values over every node and every round. Interestingly, in regular graphs the rotor walk and the random walk model can be viewed as discrete and continuous load balancing algorithms.

By adopting this view, we obtained bounds on the deviation of the rotor walk model from the continuous random walk model. Here, our main contribution is to prove that

Table 1.1: The Notation Used Throughout the Thesis

n	Number of processors
V	$V = \{1, \ldots, n\}$; the set of processors
E	The set of edges denoting the underlying connections of the processors
$G = (V, E)$	A simple undirected loop-free graph denoting the network's topology with V as its nodes and E as its edges
$N(i)$	The set of direct neighbours of the node i
d_i	$d_i = \lvert N(i)\rvert$; Degree of node i
d	Maximum node degree
s_i	Speed of node i; we assume $s_1 = 1$, and for all $2 \leqslant i \leqslant n$, we have $s_i \geqslant 1$.
s	$s := s_1 + \cdots + s_n$; capacity of the network
m	Number of tasks
w_j	Weight of task j
$w_{\max}$	$w_{\max} := \max_j w_j$; maximum weight of any task
w	$w = w_1 + \cdots + w_m$; the total weight of all the tasks
x_i	*Load* of a node i which is defined as the total weight of the tasks assigned to i
ℓ_i	$\ell_i := x_i/s_i$; *makespan* of node i which is defined as its load divided by its speed
$\bar{x}_i$	$\bar{x}_i := s_i \cdot w/s$; the desired load of the node i
$\bar{x}$	$\bar{x} := m/n$; the common value for $\bar{x}_i$ in a homogeneous networks
$x^{\mathcal{P}}(t)$	$x^{\mathcal{P}}(t) := (x_1^{\mathcal{P}}(t), \ldots, x_n^{\mathcal{P}}(t))$; the load vector at the beginning of round t of process $\mathcal{P}$. When clear from the context, the process superscript may be omitted.
$y_{i,j}^{\mathcal{P}}(t)$	the load transferred from node i to node j during the round t of process $\mathcal{P}$. When clear from the context, the process superscript may be omitted.
T	Convergence time of various processes, depending on the context
$\{a\}$	For $a \in \mathbb{R}$, we define $\{a\} := a - \lfloor a \rfloor$

Table 1.2: Terms Used Throughout the Thesis

Continuous process	A process in which tasks may be broken into smaller parts arbitrarily before being transferred
Discrete process	A process that only transfers tasks atomically without breaking them into smaller parts
Homogeneous network	A network of processors with identical speeds
Heterogeneous network	A network of processors with different speeds
Token	A single load unit in the uniform task model
Discrepancy of a vector v	$\max_i v_i - \min_i v_i$
Discrepancy	Makespan discrepancy; the discrepancy of the makespan vector
Load Discrepancy	Discrepancy of the load vector; for homogeneous networks this is equal to the makespan discrepancy.
Convergence time of a process	The number of rounds a process needs to reach a state with constant discrepancy
w.h.p.	An event occurs *with high probability (w.h.p)* if its probability is at least $1 - \mathcal{O}(n^{-\alpha})$ for some constant $\alpha > 0$.

viewed as a discrete load balancing algorithm, the rotor walk model exhibits the *bounded-error property* on any graph (Theorem 4.1). Hence, similar to the process of [41], we get deviation bounds of $\mathcal{O}(\log^{3/2} n)$ for hypercubes and $\mathcal{O}(1)$ for constant-degree tori. Since our analysis is based on the analysis of [41], due to the same technical limitations in that paper we were not able to obtain improved bounds for other graph classes. In Section 4.2, we analyze a randomized rotor walk model, as a discrete load balancing scheme with randomized rounding. This algorithm still distributes tokens in a round-robin fashion, with the slight change that token allocation in each each round starts from a random neighbour (in contrast, the original deterministic algorithm which continues the allocation based on which neighbour got the token in the previous round). Though this new algorithm uses much fewer number of random bits compared to the random walk model and also previous randomized discrete load balancing algorithms [10, 41], it achieves deviation bounds of $\mathcal{O}(d \log\log n/(1-\lambda))$ for regular graphs, $\Theta(\log n)$ for hypercubes, and $\mathcal{O}(\sqrt{\log n})$ for tori. These bounds are at least as good as those obtained in [10, 41] (see Table 4.2).

In Chapter 5 we introduce an abstract view — defined in the following – of a randomized discrete process[2].

We note that each balancing process A can be regarded as a function that, given the current state of the network, determines for every edge e and round t the amount of load that has to be transferred over e in t. Hence, we can regard $y^A(t)$ as the result of applying a function A, i.e. $y^A(t) = A(x^{\mathcal{A}}(t))$. Using this we formally define discrete processes:

Definition 1.1. *Let C be a continuous process. A process D is said to be a discrete version of C with rounding scheme R_D if for every vector $\mathbf{x}$, we have $D(\mathbf{x}) = R_D(C(\mathbf{x}))$ where R_D is a function that rounds each entry of the matrix to an integer.*

Applying the abstract view defined above to the analysis techniques in previous works, we obtained general bounds for the case of heterogeneous networks. The above result combined with some linear algebraic analysis for the particular case of second-order diffusion, provided first results for the randomized second-order diffusion algorithm (SOS) defined by our framework.

[2]Although not all discrete algorithms fit into this definition (flow imitation is an example), the definition covers a rather large class of discrete algorithms in the heterogeneous network model.

We show that randomized SOS has a deviation of $\mathcal{O}\left(\frac{d\cdot\log s_{\max}\cdot\sqrt{\log n}}{(1-\lambda)^{3/4}}\right)$, where λ is the second largest eigenvalue and $s_{\max}$ is the maximum speed.

In second-order processes a node may attempt to send out more load than its available load. To address this issue, we provide bounds on the minimum load of any node which is sufficient to prevent such conditions. We show that the continuous SOS scheme will not generate negative load if (at $t = 0$) the minimum load of every node is at least $\mathcal{O}\left(\sqrt{n}\cdot\Delta(0)/\sqrt{1-\lambda}\right)$. Here $\Delta(0)$ is the difference between the minimum and maximum load at $t = 0$. For discrete SOS schemes we show a bound of $\mathcal{O}\left(\sqrt{n}\cdot\Delta(0)+d^2/\sqrt{1-\lambda}\right)$. To the best of our knowledge these are the first results specifying the sufficient minimum load to avoid negative load.

Chapter 2

Related Work

There is a vast amount of literature about load balancing. In this chapter, we first give an overview of the results on continuous (Section 2.1) and discrete neighbourhood load balancing (Section 2.2). When not stated otherwise, the results are for the uniform case without speeds and weights. After that we explore farther related models in Section 2.3.

2.1 Continuous Load Balancing

The first diffusion algorithm (also called *first order schedule*, FOS) was independently introduced by Cybenko [20] and Boillat [14]. Their results were later generalized to the case of heterogeneous networks in [29]. To introduce the FOS process we first need some additional notation. Let $x_i(t)$ be the load of resource i at the beginning of round $t \geqslant 0$ of the process. We define $x(t) = (x_1(t), \ldots, x_n(t))$ as the *load vector* in the beginning of round t. For an arbitrary node i let $N(i)$ be the set of neighbours of i. Furthermore, let d_i be the degree of node i and recall that s_i is the speed of resource i. For $j \in N(i)$ let $y_{i,j}(t)$ be the (positive) amount of load transferred from node i to node j in round t. Then the FOS process is defined as follows.

$$y_{i,j}(t) = \frac{\alpha_{i,j}}{s_i} \cdot x_i(t), \tag{2.1}$$

$$x_i(t+1) = x_i(t) - \sum_{j \in N(i)} \alpha_{i,j} \left(\frac{x_i(t)}{s_i} - \frac{x_j(t)}{s_j} \right), \tag{2.2}$$

where for $j \in N(i), \alpha_{i,j} = \alpha_{j,i}$ are positive parameters to be chosen with the restriction that for all i, we must have $\sum_{j \in N(i)} \alpha_{i,j} < s_i$. Common choices for $\alpha_{i,j}$

are $1/(2\max(d_i, d_j))$ or $1/(\max(d_i, d_j) + 1)$. The process can also be defined using a so-called *diffusion matrix* P, where for all i we have $P_{i,i} = 1 - \sum_j \alpha_{i,j}/s_i$ and for $j \in N(i)$, we have $P_{i,j} = \alpha_{i,j}/s_i$. Then we have

$$x(t+1) = x(t) \cdot P, \tag{2.3}$$

where P is a right stochastic matrix[1] that can be viewed as the transition matrix of an ergodic[2] Markov chain with a unique steady-state distribution $(s_1/s, \ldots, s_i/s, \ldots, s_n/s)$. This holds regardless of the specific choice of $\alpha_{i,j}$'s, as long as they satisfy the mentioned constraints. Hence, repeatedly applying Equation (2.3) leads to the perfectly balanced state. Let K denote the initial discrepancy and λ the second-largest eigenvalue in absolute value of the diffusion matrix. Here and throughout, we use the variable T to denote the balancing time of various continuous processes. When not clear from the context, we redefine T explicitly. Then [29, 57, 62] use the above approach to show that

$$T = \mathcal{O}\left(\frac{\log(Kn)}{1-\lambda}\right),$$

where the result of [29] is for the case of non-uniform speeds.

Muthukrishnan et al. [57] introduced the *second order schedule* (SOS). Later, the SOS was generalized by Elsässer et al. [29] to the case of non-uniform speeds. The SOS method is inspired by a numerical iterative method called successive over-relaxation. In SOS, the amount of load transmitted over each edge depends on the current state of the network as well as the load transferred in the previous round. The first round equation is similar to FOS Equations (2.1) and (2.2), and subsequent rounds are defined by

$$y_{i,j}(t) = (\beta - 1) \cdot y_{i,j}(t-1) + \beta \cdot \frac{\alpha_{i,j}}{s_i} \cdot x_i(t), \tag{2.4}$$

where $\alpha_{i,j}$'s are as in Equation (2.1) and $0 < \beta \leqslant 2$. For some choices of β SOS converges faster than FOS [57]. The optimal choice for β is known to be $2/(1+\sqrt{1-\lambda^2})$ [29, 57], which leads to

$$T = \mathcal{O}\left(\frac{\log(Kn)}{\sqrt{1-\lambda}}\right).$$

Here again, the result of [29] is for the case of non-uniform speeds.

[1] A real square matrix, with each row summing to 1.

[2] An ergodic Markov chain is a Markov chain which is both *irreducible* (every state is reachable from every other state) and *aperiodic* (for which, a sufficient condition is to have at least one non-zero loop probability).

For SOS it can happen that the total outgoing demand $\sum_{j \in N(i)} y_{i,j}$ exceeds the load x_i, which results in so-called *negative load*.

The *dimension exchange* model which we will also refer to as the *matching based* model is motivated by single-port architectures as opposed to the diffusion model which necessitates multi-port communication [20, 44, 45, 62]. In the matching based model every node balances its load with only one neighbour. More formally, the load transfer in each round is restricted to a – not necessarily perfect – matching of the underlying graph. Let (i, j) be an edge in the matching of round t. Then resource i and resource j calculate $y_{i,j}$ and $y_{j,i}$ such that their makespans are equalized. This can be done by the following equations, which can be regarded as a special case of the Equations (2.1) and (2.2)).

$$y_{i,j}(t) = \frac{\alpha_{i,j}}{s_i} \cdot x_i(t), \tag{2.5}$$
$$x_i(t+1) = \frac{s_i}{s_i + s_j} \cdot \big(x_i(t) + x_j(t)\big),$$

where $\alpha_{i,j} = \alpha_{j,i} = s_i s_j/(s_i + s_j)$.

Similar to the diffusion load balancing, the process can be defined using a sequence of matrices $\{P(0), P(1), \ldots\}$, where $P(t)$ represents a modification of the adjacency matrix of the matching that is used in step t, as follows. If (i, j) is in the matching of the round t then $P_{i,j}(t) = \alpha_{i,j}/s_i$ and $P_{i,i}(t) = 1 - P_{i,j}(t)$. If i is not matched then $P_{i,i}(t) = 1$. All other entries are zero.

Several publications assume that a fixed set of matchings (usually roughly maximum degree many) is given and the matchings are used periodically. Hence, for all t we have $P(t) = P(t \bmod \tilde{d})$ where $\tilde{d}$ is the length of the period. The model was originally introduced in [49], together with a distributed edge-colouring algorithm (see also [58, 59]) that can be used to construct the matchings. As far as we know, the algorithms in the matching model have been analyzed only for the case of uniform tasks and speeds. The analysis for the algorithms using periodic matchings is very similar to the analysis of FOS. The convergence was first analyzed in [20] for hypercubes and in [50] for arbitrary graphs. Define

$$\mathcal{P} := P(0) \cdot P(1) \cdot \ldots \cdot P(\tilde{d} - 1)$$

and let λ be the second-largest eigenvalue in absolute value of the matrix $\mathcal{P}$ [3]. If we

[3]For the case where the matrix $\mathcal{P}$ is not symmetric, we need to define λ as the second-largest eigenvalue in absolute value of the matrix $\mathcal{P} \cdot \mathcal{P}^T$ (see the discussion in [62, Section 3] for more details).

consider a group of $\tilde{d}$ consecutive rounds together, we have $x((t+1)\cdot\tilde{d}) = x(t\cdot\tilde{d})\cdot\mathcal{P}$, which is similar to the Equation (2.3) of the first order diffusion. Consequently, we have [62]

$$T = \mathcal{O}\left(\frac{\tilde{d}\cdot\log(Kn)}{1-\lambda}\right).$$

Another approach is to use in every step t a randomly generated matching. In [44], it is shown that w.h.p.

$$T = \mathcal{O}\left(\frac{d\cdot\log(Kn)}{\gamma}\right),$$

where $\gamma \leqslant d$ is the second-smallest eigenvalue of the Laplacian matrix of the original graph G.

2.2 Discrete Load Balancing

As far as we know, existing papers consider only discrete algorithms in the uniform task model. Many publications consider uniform processors [10, 9, 27, 28, 41, 39, 44, 57, 62, 64], while a few others incorporate processor speeds into the model [2, 30]. There are two main approaches for analyzing discrete neighbourhood load balancing processes. The first one is to use potential functions and the second one is to compare the performance of a discrete process with that of a continuous version of that process.

The first approach is used in [44] for the random matching model and in [57] for the diffusion model. In both papers the discrete process calculates the amount of load to be transmitted over every edge as in the continuous process (as in Equations (2.1) and (2.2)), which is then rounded down. The potential function used in [44, 57] is defined as follows,

$$\Phi(t) := \sum_{i\in V}\left(x_i(t) - \frac{m}{n}\right)^2 \tag{2.6}$$

It is shown in [57] that in every round of the continuous FOS, $\Phi(t)$ drops at least by a multiplicative factor of λ^2. Furthermore, in the discrete process, for any choice of parameter $\epsilon < 1$, if $\Phi(t) \geqslant 16\,d^2n^2/\epsilon^2$ then $\Phi(t+1) \leqslant (1+\epsilon)\cdot\lambda^2\cdot\Phi(t)$. Thus, the discrete process mostly behaves similarly to the continuous process as long as the potential is large enough. More precisely, the authors of [57] show that the potential is reduced to

$$\mathcal{O}\left(\frac{d^2n^2}{\epsilon^2}\right) \text{ within } \mathcal{O}\left(\frac{\log\Phi(0)}{1-(1+\epsilon)\lambda^2}\right) \text{ rounds.} \tag{2.7}$$

For FOS schemes, [57] left it as an open question to analyze the potential drop when the potential is smaller than $\mathcal{O}(d^2n^2)$. Ghosh and Muthukrishnan [44] consider the random matching model with the same potential function as defined in Equation (2.6). They show that as long as $\Phi(t) = \Omega(dn)$, the potential drops by at least a multiplicative factor of $\Omega(\gamma/d)$ per round, where γ is the second smallest eigenvalue of the Laplacian of the graph. For a smaller potential they were still able to show an additive drop of $\Omega(1/d)$ per round. Using this fact, they prove that the max-min discrepancy is reduced to

$$\mathcal{O}(\text{diam}(G)) \text{ within } \mathcal{O}\left(\frac{d \log \Phi(0) + d^2 n}{\gamma}\right) \text{ rounds,}$$

w.h.p. Note that analyzing algorithms in the matching model is easier since in this model the potential never increases.

The discrete SOS process was first analyzed in [27]. The authors of that paper measure the distance to the balanced state by the second norm of the difference between the load vector and the balanced vector. This measure is equal to $\sqrt{\Phi}$, where Φ is defined in Equation (2.6). The authors show that in sufficiently many rounds $\sqrt{\Phi(t)}$ is reduced to $\mathcal{O}((d \cdot \sqrt{n})/(1-\lambda))$.

The above results for FOS and SOS are generalized in [30] to the case of non-uniform speeds. Note that for non-uniform speeds, the balanced allocation is $(w/s) \cdot (s_1, \ldots, s_n)$, where $w := w_1 + \ldots + w_m$ is the total task weight and $s := s_1 + \ldots + s_n$ is the total speed of the processors. Hence, the optimal load of processor i is $s_i \cdot w/s$, and the potential function defined in Equation (2.6) becomes

$$\Phi(t) := \sum_{i \in V} \left(x_i(t) - s_i \cdot w/s\right)^2 .$$

In [30] the authors measure the distance to the balanced state by the second norm of the difference between the load vector and the balanced load vector $(s_1, \ldots, s_n)w/s$. They show that after a sufficient number of rounds t, $\sqrt{\Phi(t)}$ is reduced to $\mathcal{O}((d \cdot \sqrt{n \cdot s_{\max}})/(1-\lambda))$ in an FOS or SOS process.

The second analysis technique was introduced by Rabani et al. [62] and was later used in [10, 41, 39, 64]. In [62], the authors introduce a framework to analyze a large class of discrete neighbourhood load balancing algorithms. They transform a continuous algorithm A_c into a discrete version A_d that tries to stay as close to A_c as possible. Assume we apply both algorithms on a load vector x. Then, for every edge e, both A_c and A_d calculate the amount of load y_e that A_c would send over e. A_c will send

exactly y_e tokens over e and A_d will *round down* y_e to an integer y'_e. The difference between y_e and the amount of load y'_e that is transferred in reality is called *rounding error*, which we denote by

$$\Delta y_e := y_e - y'_e \tag{2.8}$$

They show that the max-min discrepancy at time T is bounded by

$$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right).$$

See Tables 2.1 and 2.2 for the results of [62] for different graph classes. Their analysis framework applies to both FOS and dimension-exchange processes, and the results hold for a wide class of rounding schemes, as long as the rounding errors are bounded by a constant.

Another technique using the potential function of Equation (2.6) is proposed in [9], where the potential drop is estimated using a *sequentialization* technique. In this technique, all the edges are assigned weights proportional to their scheduled load transfer. To estimate the potential drop, they consider a sequentialized version of the process in which edges are activated sequentially in increasing order of weight. Berenbrink et al. [9] show that under certain conditions the potential drop of the sequentialized and the original FOS process differ only by a constant factor. They show that

$$\Phi(t) = \mathcal{O}(d^3 \cdot n/\gamma) \text{ within } \mathcal{O}\left(\frac{d}{\gamma} \cdot \log\left(\frac{\Phi(0) \cdot \gamma}{d^3 \cdot n}\right)\right) \text{ rounds.}$$

Adolphs and Berenbrink [2] present a potential function based analysis of FOS for resources with non-uniform speeds. The analysis has two steps. First the authors show that

$$\Phi(t) = \mathcal{O}\left(d^3 \cdot s_{\max}^3 \cdot \frac{n}{\gamma}\right)$$
$$\text{after } \mathcal{O}\left(\frac{d \cdot s_{\max}^2 \log(m/n)}{\gamma}\right) \text{ rounds.}$$

In comparison with the result of [57] (see Equation (2.7)), they use fewer rounds when m is small compared to n and the potential bound is better for graphs with good expansion. In the second step, they use a different potential function to show that the max-min discrepancy is reduced to

$$\mathcal{O}(\operatorname{diam}(G) \cdot d \cdot s_{\max})$$
$$\text{after } \mathcal{O}\left(\frac{n \cdot d^3 \cdot s_{\max}^3}{\gamma}\right) \text{ additional rounds.}$$

Both [9] and [2] employ techniques from spectral graph theory by representing the potential drop as a function of a quadratic from.

There are also several results showing lower bounds. When the continuous flow is rounded down, the final discrepancy is $\Omega(d \cdot \mathrm{diam}(G))$ for a discrete FOS process [41, 44] and $\Omega(\mathrm{diam}(G))$ for a discrete process in the matching model [44]. Friedrich and Sauerwald [39] consider the matching model and show that for any graph the discrepancy is at least $\Omega(\log n / \log\log n)$ after $\mathcal{O}(\log n)$ rounds if the rounding decisions and the initial load distribution are determined by an adversary.

2.2.1 Improved Processes for Discrete Load Balancing

The next three subsections discuss three different approaches that were used in order to reduce the difference (caused by the rounding error) in the load distribution between discrete and continuous balancing processes.

Random Walk Approach. Algorithms that use this approach [27, 30, 28] are developed in the uniform task model. These algorithms consist of two phases. The first phase uses the discrete diffusion approach of [62]. In the second phase it is assumed that the nodes know the average load m/n (which can be easily obtained by simulating the continuous diffusion algorithm on any node in Phase 1).

The second so-called fine balancing phase is not a simple neighbourhood load balancing strategy. Every token above $\alpha = m/n + c$ is marked as a *positive token*, and nodes with fewer than α tokens generate a *negative token* for every *hole* they have. The negative and positive tokens then perform one random walk step in each round. In reality, the movement of a negative token from node i to node j is translated as a token movement from node j to node i. Hence, whenever a negative token hits a positive token, both are eliminated. Note that this method can create negative loads when in one round too many negative tokens move to the same node. The tightest analysis of the algorithm is due to Elsässer and Sauerwald [28] where the authors achieve constant max-min discrepancy in $\mathcal{O}(T)$ rounds.

Randomized Rounding. This technique applies randomized rounding on $y_{i,j}$ in order to improve the bounds on the discrepancy for discrete neighbourhood load balancing. It was suggested in [66] and first analyzed by [39]. In the latter paper the authors consider a discrete dimension exchange algorithm for the matching model. Every node i that

is connected to a matching edge (i, j) first calculates $y_{i,j}$ as in Equation (2.5). If that value is positive, it rounds it up or down, each with a probability one half. The authors combine the approach of [62] with analysis techniques for randomized algorithms to show improved discrepancy bounds for general graphs. For expanders, they provide a separate analysis that shows constant discrepancy is achieved by slightly increasing the running time. See Table 2.2 for a detailed statement of their results.

For arbitrary regular graphs the results of [39] were further improved in [64]. One of the main ideas of this paper is to show *negative dependence* [25] among the token locations, which enables them to use simple Chernoff bounds for their proofs. They show that a constant final discrepancy can be achieved within $\mathcal{O}(T)$ rounds for regular graphs in the random matching model, and constant-degree regular graphs in the periodic matching model. Note that the constant hidden in the asymptotic notation is large and the bounds can be achieved with probability $1 - \exp(-\log^c n)$ for some constant $c < 1$. See Table 2.2 for more detailed results.

A randomized rounding FOS process is considered in [41]. Similar to [39, 64] the process rounds $y_{i,j}$ up or down, with a probability such that the expected load forwarded over edge (i, j) is exactly $y_{i,j}$. Again, the analysis is based on the framework of [62]. See Table 2.1 for a detailed statement of the results. Note that, if a node rounds up for too many edges, it may not have sufficiently many tokens. This can lead to negative load on some of the nodes. In contrast, Berenbrink et al. [10] propose an FOS process that uses randomized rounding but avoids this problem. Again, every node i calculates for every edge (i, j) the amount of load $y_{i,j}$ that would be transferred in the continuous case (see Equation (2.1)) and rounds it down. The remaining $\sum_{j \in N(i) \cup \{i\}} (y_{i,j} - \lfloor y_{i,j} \rfloor)$ so-called *excess tokens* are then distributed as follows. Instead of rounding $y_{i,j}$ for every edge, node i forwards its excess tokens to randomly chosen neighbours (without replacement).

Note that it is possible to get similar results if the excess tokens are sent to neighbours chosen randomly with replacement or if the neighbours are chosen in a round-robin fashion with a random starting point [4]. The max-min discrepancy results of the randomized algorithms in [10, 41] are further improved by applying the results from [64], where tighter bounds are obtained for certain graph parameters used in discrepancy bounds of [10, 41]. See Table 2.1 for more detailed statement of the results.

Deterministic Rounding. Friedrich et al. [41] follow up on the rounding idea suggested in [66], raising the question whether this randomized algorithm can be derandomized without sacrificing its performance. Instead of randomized rounding, they use a deterministic rounding scheme. For each edge (i, j) they define the *accumulated rounding error* at the end of round t as

$$\widehat{\Delta y}_{i,j}(t) := \sum_{\ell=1}^{t} \Delta y_{i,j}(\ell),$$

where the error $\Delta y_{i,j}(\ell)$ is defined in Equation (2.8). Then, in round $t+1$ each node i calculates

$$\min\left\{\left|\widehat{\Delta y}_{i,j}(t) + y_{i,j} - \lfloor y_{i,j} \rfloor\right|, \left|\widehat{\Delta y}_{i,j}(t) + y_{i,j} - \lceil y_{i,j} \rceil\right|\right\}.$$

If the first term is the minimum, node i sends $\lfloor y_{i,j} \rfloor$ many tokens over (i, j), otherwise it sends $\lceil y_{i,j} \rceil$ many tokens. First the authors show that their process has the property that for every edge and in every round the accumulated rounding error is bounded by a constant, which they call the *bounded-error property*. Then they show that for hypercubes and tori[4], any process with bounded-error property achieves final discrepancy of $\mathcal{O}(\log^{3/2} n)$ and $\mathcal{O}(1)$, respectively. Note that this algorithm might also create negative load on some of the nodes.

[4]Torus graphs (tori) are grids with same side length and wrap-around edges in all dimensions (e.g. two-dimensional torus is a doughnut-shaped graph).

Table 2.1: **Final discrepancy of discrete diffusion processes for different graph classes.** The running time of each process is $T = \mathcal{O}(\frac{\log Kn}{1-\lambda})$.

Discrete Processes	**Arbitrary Graphs**	**Expanders with $d = \mathcal{O}(1)$**	**Hypercubes**	**r-dim tori $r = \mathcal{O}(1)$**
Deterministic Rounding				
Rabani et al. [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/r})$
Friedrich et al. [41] (deterministic)	–	–	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(1)$
Randomized Rounding				
Friedrich et al. [41] (randomized)	$\mathcal{O}\left(\frac{d \log \log n}{1-\lambda}\right)$	$\mathcal{O}(\log \log n)$	$\mathcal{O}(\log^2 n \log \log n)$	$\mathcal{O}(n^{1/r} \log \log n)$
Berenbrink et al. [10]	$\mathcal{O}\left(\frac{d \log \log n}{1-\lambda}\right)$, and $\mathcal{O}\left(d\sqrt{\log n} + \sqrt{\frac{\log n \log d}{1-\lambda}}\right)$	$\mathcal{O}(\log \log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$
Sauerwald and Sun [64] applied to algorithm of [10]	$\mathcal{O}(d\sqrt{d \log n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(\sqrt{\log n})$
Sauerwald and Sun [64] applied to algorithm of [41]	$\mathcal{O}(\sqrt{d \log n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$

Table 2.2: **Final discrepancy of discrete processes in the matching model.** The running time of each process is $t = T$ unless otherwise specified.

Discrete Processes	**Arbitrary Graphs**	**Expanders with $d = \mathcal{O}(1)$**	**Hypercubes**	**r-dim tori $r = \mathcal{O}(1)$**
Periodic Matchings				
Round-Down				
Rabani et al. [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/r})$
Randomized Rounding				
Friedrich and Sauerwald [39]	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$, and $\mathcal{O}\left(\sqrt{\frac{d \log n}{1-\lambda}}\right)$	$\mathcal{O}(\log\log n)$, and $\mathcal{O}(1)^\dagger$	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(n^{1/2r}\sqrt{\log n})$
Sauerwald and Sun [64]	$\mathcal{O}(\log^\varepsilon n)^*$, and $\mathcal{O}(\log\log n)^\P$	$\mathcal{O}(1)^*$	$\mathcal{O}(\log^\varepsilon n)^*$	$\mathcal{O}(1)^*$
Random Matchings				
Round-Down				
Rabani et al. [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/r})$
Randomized Rounding				
Friedrich and Sauerwald [39]	$\mathcal{O}\left(\sqrt{\frac{\log^3 n}{1-\lambda}}\right)$	$\mathcal{O}(1)^\dagger$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/2r}\log n)$
Sauerwald and Sun [64]	$\mathcal{O}(\log^\varepsilon n)^*$, and $\mathcal{O}(\log\log n)^\P$	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$	$\mathcal{O}(1)^*$

* Unlike other probabilistic bounds that hold with probability $1 - n^{-\Omega(1)}$, these bounds hold with probability $1 - \exp(-(\log n)^c)$, for some $c \ll 1$

† in $t = \mathcal{O}(T \cdot \log^3 \log n)$ rounds

¶ in $t = \mathcal{O}(T \cdot \log\log n)$ rounds

2.3 Other Related Models

2.3.1 The Game-Theoretic Model

In the models we have discussed so far, each processor has control over its tasks and can decide where to send each of the tasks. In contrast, in the game theoretic model – which we will refer to as the *network load balancing game* – tasks are autonomous selfish agents that can decide independently to which neighbor to migrate. The decisions are made based on the *cost* of residing on the processors which are non-decreasing functions of the processors' load (or simply their makespan). The aim is to design a mechanism by following which agents migrate from overloaded to underloaded resources until the allocation is more-or-less balanced. Two closely related models are *congestion games* [63] and *job scheduling games* [54]. The two models (which are equivalent in some special cases) are defined below:

Definition 2.1 (Congestion Game)**.** *A congestion game consists of a set E of n facilities, and m players where each player i has a set $s_i \subseteq 2^E$ of possible strategies*[5]*. Each facility e, has a positive non-decreasing cost function $d_e : \mathbb{N} \to \mathbb{R}$. For a strategy vector $(P_1, P_2, \ldots, P_m)$, load of a facility e is $x_e := |\{i : e \in P_i\}|$ and the cost experienced by each player i is $\sum_{e \in P_i} d_e(x_e)$.*

When the graph is complete and tasks and processors are identical, the network load balancing game becomes a special kind of *singleton congestion game* [51] where for all i, s_i is the set of all singleton subsets of facilities (processors).

Definition 2.2 (Job-Scheduling Game[6])**.** *In a job-scheduling game, a set of m jobs (players) $\mathcal{J} = j_1, j_2, \ldots, j_m$ is to be assigned to a set of n machines $\mathcal{M} = M_1, \ldots, M_n$, using an assignment or schedule function $\mathcal{A} : \mathcal{J} \to \mathcal{M}$. Each job j_k has size w_k and each machine M_i has speed s_i. The running time of j_k on a machine of speed s is w_k/s. The completion time of the machine M_i is $\sum_{k:\mathcal{A}(j_k)=M_i} w_k/s_i$. The cost experienced by each player is the completion time of the machine on which this job is running.*

In complete graphs, network load balancing games are equivalent to job-scheduling games.

---

[5] *A player's strategy is the subset of facilities the player chooses.*

[6] Here we define the job-scheduling on *uniformly related machines*. The general job-scheduling game considers *unrelated machines* model which means that there is an $m \times n$ matrix indicating the processing time of job i if executed on machine j.

In both of the models defined above, the social cost of a strategy vector (schedule) is the maximum cost experienced by any player. Next, we outline the existing results in both areas, focusing on the results they imply in the area of network load balancing games (hence, only for complete graphs). After that we mention results in the area of network load balancing games (general graphs). When considering a network load balancing game, there are two important factors to be analyzed: first, how far the equilibria (if there is any) could be from the balanced state; and second, the time to reach an equilibrium (convergence time).

Quality of Equilibria. Rosenthal [63] showed that congestion games are potential games and therefore possess pure Nash Equilibria (NE). The potential function used in [63] was $\Phi = \sum_{e \in E} \sum_{k=1}^{x_e} d_e(k)$.

The criterion to measure the quality of equilibria is *price of anarchy* (POA) which is the worst case ratio between the cost of a pure Nash equilibrium and the cost of an optimal schedule. As Czumaj and Vöcking [21] showed, in job-scheduling games on uniformly related machines, the POA is $\Theta(\log n / \log \log n)$. Feldmann et al. [36] proved that the POA for $n = 2$ and $n = 3$ are $(\sqrt{5}+1)/2$ and 2, respectively. In [31], the exact POA for two machines is obtained as a function of machine speeds. For n identical machines, it can be deduced from the results of [65, 37] that the POA is $2n/(n+1)$ [32].

Convergence Rate. Here, we first consider centralized algorithms for finding the equilibria, and then move to distributed algorithms designed to reach equilibria.

Convergence in the Centralized Model. In congestion games, Fabrikant et al. [35] show that finding pure Nash equilibria is generally PLS-complete, but for *symmetric network congestion games* [7] polynomial-time algorithms exist through a reduction to min-cost flow. For *matroid congestion games*[8], Ackermann et al. [1] show that best-response sequences have length polynomial in the number of players, resources, and rank of the matroids (which is 1 in case of a load balancing game). Ieong et al. [51] give polynomial bounds for best-response and better-response sequences in singleton congestion games. All these results can be regarded as convergence time bounds in a "sequential" model called the *elementary step system* (ESS). In ESS, at most one move

[7] A class of congestion games in which facilities represent different *paths* between a fixed pair of nodes in a network. Load balancing games on complete graphs falls into this class.

[8] A class of congestion games where each s_i consists of the bases of a matroid over the set of resources

is performed in each step. Chien and Sinclair [15] study a version of the ESS in the context of approximate NE, and show that in some cases the ϵ-Nash dynamics may find an ϵ-NE where finding an exact NE is PLS-complete.

For job-scheduling games, it is shown in [38] that pure Nash equilibria can be found in polynomial time, while computing a pure Nash equilibrium with minimum social cost is NP-hard. There is, however, a PTAS for approximating the cost of the best equilibrium [42]. Feldmann et al. [36] show how to find in polynomial-time, a sequence of (not necessarily selfish) moves that lead to a Nash equilibrium. For identical machines, they give exponential lower bound of $\mathcal{O}(2^{\sqrt{m}})$ and upper bound of $\mathcal{O}(2^m - 1)$ on the length of the best-response sequences. If additionally job weights are integers, any sequence of selfish moves converges in $\mathcal{O}(w + m)$ moves where $w = \sum_{i \in \mathcal{J}} w_i$.

The choice of which job to move next has a significant impact on the convergence time. For identical machines, it has been shown [34] that the number of moves made by an algorithm that always moves the lowest-weight task, may be exponential while an algorithm in which higher-weight tasks are favoured and performs best-response moves is guaranteed to converge within m steps. If on the other hand, the next job is selected from among the candidates at random or in a FIFO manner, the (expected) length of the best-response sequence becomes of $\mathcal{O}(m^2)$ [34]. In the general case of machines with speeds, convergence time to ϵ-Nash equilibrium[9] is known to be of $\mathcal{O}(w^2 s_{\max}^2/\epsilon)$ where $s_{\max}$ is the maximum speed [34].

Convergence in Distributed Control. The convergence rate results mentioned above are centralized solutions. However, implementing a centralized controller is usually difficult and undesirable in a truly distributed system. This gives a strong motivation to think of distributed algorithms. For the identical machine model, a distributed algorithm is given in [46] in which each job repeatedly samples a random neighbor to perform a selfish move after delay periods of random length. Assuming the continuous time model, Goldberg [46] shows that the algorithm's behavior is similar to a centralized algorithm that randomly chooses the next job to move. This leads to an expected sequence length of $\mathcal{O}(w_{\max}^2 n^4 m^5 \log(mn))$ where $w_{\max}$ is the maximum job weight. When $w_{\max} \leqslant m$, this bound reduces to $\mathcal{O}(n^2 m w_{\max})$. Furthermore, for $n = 2$, a tight bound of $\Theta(m^2)$ is obtained [46]. Although distributed, the algorithm in [46] follows the ESS model which

[9] The system is in ϵ-Nash equilibrium when no job can benefit more than ϵ from unilaterally migrating to another machine.

allows at most one user to reroute in each stage. The convergence time is therefore $\Omega(n)$. This is an efficiency drawback since networks normally support concurrent moves.

Even-Dar and Mansour [33] identify two types of equilibria that should be present in concurrent job-scheduling games (job scheduling games which work in synchronous rounds). First, the Nash equilibrium which the algorithm should eventually converge to. Second, a Nash equilibrium among the users' rerouting policies which should be present in all rounds. In other words, no greedy user should have any incentive to deviate from her policy. Such set of policies is referred to as *Nash rerouting policies* [33]. Even-Dar and Mansour [33] consider synchronous concurrent decisions where jobs migrate probabilistically from overloaded to underloaded resources with probabilities proportional to deviation of destination's load from its target value. The expected convergence time is $\mathcal{O}(\log\log m + \log n)$. This logarithmic convergence holds for a general class of rerouting algorithms that among other properties, support jumps only from overloaded to underloaded resources. Hence, the logarithmic convergence time comes at the price of requiring a relatively big amount of global knowledge because in each round, a job on an overloaded resource is required to know which resources are overloaded.

Berenbrink et al. [8] consider two protocols for the case of identical resources and agents in which jobs only need to know their current resource load and the load of a randomly chosen alternative. In the first (resp. second) protocol, each agent residing on resource i examines the load of a uniformly sampled resource j. If $x_i > x_j$ (resp. $x_i > x_j + 1$), the agent moves from i to j with probability $1 - x_j/x_i$. The second protocol is different from the first in which it does not allow *neutral moves*[10]. Their analyzes show that for $m^{1/3} \geqslant n$, both protocols converge to an ϵ-NE[11] in expected time $\mathcal{O}(\log\log m)$. Moreover, the second protocol reaches an NE in expectedly $\mathcal{O}(n^4)$ many additional rounds, while for a particular initial configuration, the first protocol expectedly needs about $\exp(\Theta(\sqrt{n}))$ rounds to reach an NE. Therefore, disallowing neutral moves results in fast convergence from an ϵ-NE to NE. For both protocols, carefully chosen initial states require expectedly $\Omega(\log\log m + n)$ many rounds to converge to an NE. A similar protocol has been considered in [11] for weighted jobs, in which job b moves from the machine i to the machine j with probability $\rho(1 - x_j/x_i)$ if $x_i \geqslant x_j + (1+\epsilon)w_b$. Here, $\rho = \epsilon/8$ is a

[10] Neutral moves occur when strategy changes do not change the cost.

[11] Here, ϵ-NE is a state in which no user can reduce her cost by a multiplicative factor of less than $1 - \epsilon$ by unilaterally changing strategy.

slowdown factor. The expected convergence time to an ϵ-NE is $\mathcal{O}(n \cdot m \cdot w_{\max}^3 \epsilon^{-2})$. In the special case of identical jobs and $\epsilon = 1$, an NE – which corresponds to a perfectly balanced state as in [8] – is reached in expectedly $\mathcal{O}(\log m + n \log n)$ and $\Omega(\log m + n)$ rounds. The slowed-down protocol has faster convergence compared to the protocol in [8] only when the state is close to equilibrium. Therefore, as shown in [11], by combining the two protocols in two phases, a faster protocol with expected convergence time of $\mathcal{O}(\log \log m + n \log n)$ can be obtained.

Convergence Results for General Graphs. The results mentioned so far in the area of congestion games and job-scheduling games can imply results for the network load balancing game only on complete graphs. The reason is in these two models one can move from any processor or resource to another. The first paper to consider the general case of arbitrary network topology and heterogeneous jobs or machines was [12]. By potential analysis, they obtain bounds on the expected time to achieve exact or approximate NE's. The potential function used is $\Phi_r(t) := \sum_{i \in V} x_i^{\mathcal{RD}}(t) \cdot (x_i^{\mathcal{RD}}(t) + r)/s_i$. They show an expected multiplicative potential drop, to bound the convergence time to an approximate Nash equilibrium. From there, a constant drop is used to achieve a Nash equilibrium. [3] improved the bounds of [12] by analyzing the same potential function. The obtained expected convergence time bounds are polynomial in $d, s_{\max}$, and $1/(1-\gamma)$ for approximate/exact NE. There is also a factor of $\log(m/n)$ for the convergence time to an approximate NE. For additional rounds leading to an exact NE, the extra factor is linear in n.

2.3.2 The Balls-into-Bins Model

In a balls-into-bins process, m balls are thrown into n bins, each ball landing in a bin chosen independently and uniformly at random [56]. There are several interesting random variables one may want to estimate, such as the number of empty bins and the maximum number of balls in any bin. In general one might be interested to know how the distribution of bin loads looks like.

Many of these questions are frequently revisited in design and analysis of algorithms. For instance, the birthday paradox states that for $m = \Omega(\sqrt{n})$, w.h.p., there exists a bin with more than one ball. Also, according to the coupon collector's problem, for $m = \Theta(n \log n)$, w.h.p. no bin remains empty.

Considering the tasks as balls and the resources as bins, some variants of the distributed load balancing problem meet the balls-into-bins model. For instance, in the process analyzed in [64], after mixing time many steps each token ends up on a random node with probability distribution close to uniform. Although not independent, for a fixed node the Bernoulli random variables that indicate whether or not each token ends up on that node are negatively associated. Hence, many balls-into-bins results on the maximum load carry over [64]. In this context, the most relevant balls-into-bins results are bounds on the maximum bin load.

It is well-known that the maximum load is $(1+o(1))\cdot \log n/\log\log n$ when $m = n$ [6] and $m/n+\Theta(\sqrt{m\log n/n})$ when $m \geqslant n\log n$, w.h.p. Tighter bounds have been obtained in [61] specifying the precise deviation from the mean in different cases of m as a functions of n. In a more general model of balls-into-bins [22, 6, 67], each ball is placed in the least loaded of d randomly chosen bins (ties broken arbitrarily), allocating one ball after the other. For this model, it is known that for $d \geqslant 2$ the maximum load becomes $\Theta(m/n) + (1 + o(1))\ln\ln n/\ln d)$ w.h.p [22, 6].

2.3.3 Propp Machines and Rotor-Router Walks

Originally introduced in [60], the rotor walk model was employed by Jim Propp for derandomizing the random walk, thereby frequently appearing under the names of *Propp machines* and *deterministic random walks* [17, 19, 24, 40, 52]. The *rotor walk* model works as follows: Given a graph along with an arbitrary distribution of tokens over its nodes, the process works in rounds and in every *round* each node distributes all of its tokens to its neighbours. We assume that every node is equipped with a rotor and its neighbours are arranged in a fixed circular list. One token is allocated to the neighbour indicated by the rotor, then the rotor is directed to the next neighbour in the circular list, and so on, until no token remains. The idea of the rotor walks is to get a token distribution that is similar to that resulting when the tokens perform independently random walks, while avoiding the high variance of random walks.

It has been shown that the rotor walks capture the average behaviour of random walks in a variety of respects such as hitting probabilities and hitting times [48]. Early works consider the single-walker model where only one token walks over a graph $G = (V, E)$ where V is the set of nodes and E is the set edges of the graph. This includes results on derandomizing random walks in the context of *graph exploration* [7, 13, 43, 68, 69, 71],

where a single *agent* with only local information wants to explore the whole graph. The parameters of interest are the cover time of the graph and the traversal frequency of its edges. In this model [68, 69] showed that starting from an arbitrary configuration the agent covers all edges of the graph within $O(n|E|)$ steps. Later, Bhatt et al. [13] showed that within $O(n|E|)$ steps the agent also enters an Eulerian cycle. This bound was improved in [71] to $2|E| \cdot \mathrm{diam}(G)$ steps, where $\mathrm{diam}(G)$ is the diameter of G. Klasing et al. [53] consider parallel rotor-router walk of multiple agents on rings. They obtain upper and lower bounds on the cover time which closely resembles to that of random walks. They also show that once the system stabilizes, all the nodes of the ring are always visited by some agent every $\Theta(n/k)$ steps. This bound on the number of step the expected time between successive visits to a node. These bounds hold as long as the number of agents is less than $n^{1/11}$. The model was also analyzed in the presence of dynamic edge and pointer changes [7]. Cooper et al. [16] study two so-called *equitable strategies* for graph exploration which attempt to mimic the fairness properties of random walks with respect to the use of edges. The agent is always directed to the least often used, or the longest unused, from among the adjacent edges. The latter strategy is equivalent to the rotor-router model.

Several publications address the question of how closely rotor walks approximate the expected token distributions of random walk models. The closeness of the two models is usually measured by comparing the number of tokens a node has in the two models, taking the maximum difference between the two values over every node and every round. Note that is is assumed that initially both models start with the same token distribution. This difference will be referred to as the *deviation*[12] between the two models.

Cooper and Spencer [18] consider rotor walks on infinite grids with constant dimensions; they bound the deviation by a constant which depends on the dimension of the grid. For one and two dimensions, the constants are calculated in [17] and [24]. For infinite d-regular trees Cooper et al. [19] show that there exists an initial token distribution and direction of the rotors that results in a deviation of $\Omega(\sqrt{dt})$ for an arbitrary round t. Note that the initial token distribution and rotor directions depend on t. Kijima et al. [52] obtain a general deviation bound of $\mathcal{O}(n|E|)$ for arbitrary graphs and $1.5 \log^3 n + \mathcal{O}(\log^2 n)$ for hypercubes. The general $\mathcal{O}(n|E|)$ bound holds only for lazy

[12] In the literature of rotor-router walks, this is notion is called *discrepancy*. However, here and throughout the thesis we use the word *deviation* to avoid confusion with the discrepancy measure introduced in the load balancing literature.

random walks while the latter hypercube bound holds both in case of lazy and non-lazy random walks. Note that for expanders and constant-degree tori, the deviation bound implied by the general $\mathcal{O}(n|E|)$ bound is not better than $\mathcal{O}(n^2)$.

It should be noted that the concept of rotor walks can be generalized to a wide class of non-standard random walks (where the neighbours are not chosen *uniformly* at random and tokens are also allowed to remain at the same node), resulting in deterministic versions of time-homogeneous Markov chains [48].

Chapter 3

Flow Imitation

In this chapter[1] we present two new discretization schemes: one is deterministic and the other is randomized.

The Deterministic Algorithm. In Section 3.3 we present Algorithm 3.1; a general framework that translates a continuous load balancing process into a discrete version. In every round the discrete algorithm *imitates* the continuous algorithm as closely as possible by trying to send the same amount of load over every edge as the continuous algorithm. To be more detailed, let $f_e^c(t)$ and $f_e^d(t)$, respectively, be the total load sent over edge e during the first t rounds of the continuous process and its discrete version. Then the discrete version of the algorithm tries to send a load of $\lfloor f_e^c(t) - f_e^d(t-1) \rfloor$[2] over e in round t. Note that it might not be possible for the discrete algorithm to send the required amount of load over all the edges since the load of the nodes might not be sufficiently large. In that case the node will create some dummy tokens to fulfil all demands. These dummy tokens will be regarded as "normal" tokens for the rest of the balancing process. At the end of the balancing process they will be eliminated, which reduces the makespan of the resources who were holding some of the dummy token at the end of the process. See Algorithm 3.1 for details.

Recall from Section 1.1 that *max-min discrepancy* is defined as the difference between

[1] A preliminary version of the material in this chapter was published in the *proceedings of the 2012 ACM symposium on principles of distributed computing* (PODC'12) [5]. The paper was also one of the PODC papers chosen for publication in the *Distributed Computing* journal (DIST).

[2] The discrete version of the algorithm has to know the continuous flow $f_e^c(t)$ for every edge $e = (u, v)$. This knowledge is easy to gather by simulating the continuous load balancing process in parallel on every node.

the maximum and minimum makespan and the *max-avg discrepancy* is defined as the difference between the maximum and the average makespan. Let T be the convergence time of the continuous process. Then in Theorem 3.3 we show that the following holds for the discrete version of the continuous process using Algorithm 3.1.

(1) At time T the max-avg discrepancy is $\mathcal{O}(d \cdot w_{\max})$.

(2) If the initial makespan of any node is at least $d \cdot w_{\max}$, then the above bound holds on the final max-min discrepancy as well.

Our general framework can be applied to a wide class of continuous algorithms that we called *additive terminating* algorithms (see Definitions 3.2 and 3.3).

An additive algorithm, starting with a load vector $D = D_1 + D_2$, transmits the same amount of tasks over every edge as the sum of the amounts it would transmit in the case that it were started with D_1 and D_2. A terminating algorithm does not do any load balancing actions on a completely balanced load distribution. The class of supported algorithms includes first and second order diffusion algorithms [57], dimension exchange algorithms, and random matchings models [44] as introduced in Chapter 2. This work is the first to analyze a discretization scheme supporting such a wide class of continuous algorithms. The analysis holds for arbitrary graphs, weighted tasks and resources with speed, whereas most existing papers consider only discrete algorithms in the uniform task model. Also, except a few publications [2, 30] the majority of the previous results assume uniform resource speeds.

The Randomized Algorithm. We present another transformation based on randomized rounding (Algorithm 3.2). Again, let $f_e^c(t)$ and $f_e^d(t)$, respectively, be the total load sent over edge e during the first t rounds of the continuous process and its discrete version. Then, according to Algorithm 3.2, the discrete version of the process sends either $\lfloor f_e^c(t) - f_e^d(t-1) \rfloor$ or $\lceil f_e^c(t) - f_e^d(t-1) \rceil$ tasks over edge e in round t.

Algorithm 3.2 reaches a max-avg discrepancy of

$$\frac{d}{4} + \mathcal{O}(\sqrt{d \log n})$$

after T steps (Theorem 3.8). If the initial makespan of every node is at least $d/4 + \mathcal{O}(\sqrt{d \log n})$, then the result improves to a max-min discrepancy of $\mathcal{O}(\sqrt{d \log n})$. For large values of d these bounds improve the results of the deterministic transformation presented above.

3.1 Definitions

Recall that $y_{i,j}(t)$ represents the non-negative load transferred from node i to node j at round $t \geqslant 0$. We define $f_{i,j}(t)$ as the cumulative total load transferred from i to j by the end of the round t, which is

$$f_{i,j}(t) = \sum_{\tau=0}^{t} (y_{i,j}(\tau) - y_{j,i}(\tau)).$$

We assume $f_{i,j}(-1) = 0$. We will use $\mathcal{A}$ for a continuous algorithm and $\mathfrak{D}(A)$ for its discrete counterpart (following the transformation introduced by Algorithm 3.1 in Section 3.3 or Algorithm 3.2 in Section 3.4, depending on the context). We define

$$e_{i,j}(t) = f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t)$$

as the difference in the flow forwarded over the edge (i, j) by $\mathcal{A}$ and $\mathfrak{D}(A)$ at the end of the round t. Note that

$$e_{i,j}(t) = -e_{j,i}(t)$$

and

$$f_{i,j}(t) = -f_{j,i}(t).$$

In the following, we follow up with a few definitions to be used in the statement of our new results.

Consider a continuous process $\mathcal{A}$. For the transformations introduced by Algorithm 3.1 and Algorithm 3.2, we require initial load vectors that do not lead to *negative load* in the continuous case; that is, we need to ensure that when executing $\mathcal{A}$, the outgoing demand of a node never exceeds its available load. This notion is formalized in the definition below:

Definition 3.1. *We say* $\mathcal{A}$ does not induce negative load *on* $\mathbf{x}$ *when the following holds: If* $x^{\mathcal{A}}(0) = \mathbf{x}$*, then for all* $i \in V$ *and* $t \geqslant 0$ *we have:*[3]

$$x_i^{\mathcal{A}}(t) - \sum_{j \in N(i)} y_{i,j}^{\mathcal{A}}(t) \geqslant 0.$$

[3] *For randomized processes such as the random matchings model, the statement should hold w.h.p. over the probability space.*

Note that among the processes mentioned in Section 2.1, only SOS may induce negative load, depending on the given graph structure and load distribution.

Our framework works for *additive* and *terminating* processes, as defined below. For brevity, we will not repeat these requirements throughout in the statement of the results. A balancing process is *terminating* when starting with a balanced load vector, the net load transfer over each edge is always zero. One can see that this is a very natural property of a load balancing process.

Definition 3.2 (Terminating)**.** *We say $\mathcal{A}$ is* terminating *when for any $\ell \geqslant 0$, if $x^{\mathcal{A}}(0) = \ell \cdot (s_1, \ldots, s_n)$ then $y^{\mathcal{A}}_{i,j}(t) = 0$ for all nodes i, j and round t.*

Consider a load balancing process $\mathcal{A}$. Let $\mathbf{x}'$, and $\mathbf{x}''$ be non-negative load vectors. Let $\mathbf{x} = \mathbf{x}' + \mathbf{x}''$, and suppose we start three instances of $\mathcal{A}$ with $\mathbf{x}, \mathbf{x}'$, and $\mathbf{x}''$, and that $\mathcal{A}$ does not induce negative load on $\mathbf{x}$, $\mathbf{x}'$, or $\mathbf{x}''$. Let $\mathbf{y}_{i,j}(t)$, $\mathbf{y}'_{i,j}(t)$, and $\mathbf{y}''_{i,j}(t)$, respectively, be the amount of load transferred from i to j in the round t of the three instances of the process. Then,

Definition 3.3 (Additive)**.** *$\mathcal{A}$ is said to be* additive *if the equation $\mathbf{y}_{i,j}(t) = \mathbf{y}'_{i,j}(t) + \mathbf{y}''_{i,j}(t)$ holds for all nodes i, j and round $t \geqslant 0$ regardless of the choice of $\mathbf{x}'$, $\mathbf{x}''$.*[4]

3.2 Establishing Basic Facts

The next lemma shows that the class of additive terminating processes includes several well known existing processes.

Lemma 3.1. *The following processes as defined in Section 2.1 are both additive and terminating:*

- *First order diffusion*
- *Second order diffusion*
- *Matching-based processes (using periodic/ random matchings)*

Proof. Consider a sequence of matrices $\langle P(0), P(1), \ldots \rangle$ and $0 \leqslant \beta \leqslant 2$. Observe that all the processes listed in the observation can be described by the following general equations (see Equations (2.1), (2.4), and (2.5)):

$$y_{i,j}(0) = P_{i,j}(0) \cdot x_i(0) \tag{3.1}$$

[4] *For randomized processes, the three runs are coupled with the same sequence of outcomes.*

$$y_{i,j}(t) = (\beta - 1) \cdot y_{i,j}(t-1) + \beta \cdot P_{i,j}(t) \cdot x_i(t) \qquad \text{for } t \geqslant 1, \tag{3.2}$$

where in diffusion cases, for all $t \geqslant 0$ we have $P(t) = P$.

Proof of Additivity. Let $\mathbf{x}'$, $\mathbf{x}''$ be arbitrary load distributions. Let $\mathbf{x}(t)$, $\mathbf{x}'(t)$, and $\mathbf{x}''(t)$ denote the load vector at round t starting $\mathcal{A}$ from $\mathbf{x} = \mathbf{x}' + \mathbf{x}''$, $\mathbf{x}'$, and $\mathbf{x}''$ respectively. Let $\mathbf{y}$, $\mathbf{y}'$, and $\mathbf{y}''$ denote their corresponding load transfer variables.

By induction on t, we prove that the following hold for arbitrary nodes i, j and round t:

(a) $\mathbf{x}_i(t) = \mathbf{x}'_i(t) + \mathbf{x}''_i(t)$.
(b) $\mathbf{y}_{i,j}(t) = \mathbf{y}'_{i,j}(t) + \mathbf{y}''_{i,j}(t)$

For $t = 0$, (a) holds because of the assumptions, and consequently (b) holds by Equation (3.1).

Suppose that for some $t \geqslant 0$ both $\mathbf{x}(t) = \mathbf{x}'(t) + \mathbf{x}''(t)$ and $\mathbf{y}(t) = \mathbf{y}'(t) + \mathbf{y}''(t)$ hold for every round $\leqslant t$. We show in the following that (a) and (b) must be true for $t+1$ as well. To prove (a), we use the fact that $\mathcal{A}$ does not induce negative load on $\mathbf{x}$, $\mathbf{x}'$, or $\mathbf{x}''$ to apply Equation (1.1) and write for arbitrary i:

$$\begin{aligned}
\mathbf{x}_i(t+1) &= \mathbf{x}_i(t) - \sum_{j \in N(i)} (\mathbf{y}_{i,j}(t) - \mathbf{y}_{j,i}(t)) \\
&= \mathbf{x}'(t) + \mathbf{x}''(t) - \sum_{j \in N(i)} \left(\mathbf{y}'_{i,j}(t) + \mathbf{y}''_{i,j}(t) - \mathbf{y}'_{j,i}(t) - \mathbf{y}''_{j,i}(t)\right) \\
&= \mathbf{x}'_i(t) - \sum_{j \in N(i)} \left(\mathbf{y}'_{i,j}(t) - \mathbf{y}'_{j,i}(t)\right) + \mathbf{x}''_i(t) - \sum_{j \in N(i)} \left(\mathbf{y}''_{i,j}(t) - \mathbf{y}''_{j,i}(t)\right) \\
&= \mathbf{x}'_i(t+1) + \mathbf{x}''_i(t+1).
\end{aligned} \tag{3.3}$$

Now, we proceed to prove (b). For arbitrary nodes i, j we have:

$$\begin{aligned}
&\mathbf{y}_{i,j}(t+1) \\
&= (\beta - 1) \cdot \mathbf{y}_{i,j}(t) + \beta \cdot P_{i,j}(t) \cdot \mathbf{x}_i(t+1) \\
&= (\beta - 1) \cdot \left(\mathbf{y}'_{i,j}(t) + \mathbf{y}''_{i,j}(t)\right) + \beta \cdot P_{i,j}(t) \cdot \left(\mathbf{x}'_i(t+1) + \mathbf{x}''_i(t+1)\right) \qquad (3.4) \\
&= (\beta - 1) \cdot \mathbf{y}'_{i,j}(t) + \beta \cdot P_{i,j}(t) \cdot \mathbf{x}'_i(t+1) + (\beta - 1) \cdot \mathbf{y}''_{i,j}(t) + \beta \cdot P_{i,j}(t) \cdot \mathbf{x}''_i(t+1) \\
&= \mathbf{y}'_{i,j}(t+1) + \mathbf{y}''_{i,j}(t+1)
\end{aligned}$$

where Equation (3.4) follows from Equation (3.3) and the induction hypothesis.

Proof of being Terminating. Suppose for some ℓ we have $x_i(0) = s_i \cdot \ell$ for every node i. We prove inductively that $y_{i,j}(t) = y_{j,i}(t)$ and $x(t+1) = x(0)$ for all i, j, and t. In case of the matching model, we only need to consider the matching edges. For the remaining of the proof, $\alpha_{i,j}$ is defined as in Equations (2.1), (2.4), and (2.5), depending on the process. The only property of $\alpha_{i,j}$'s we rely on here is that $\alpha_{i,j} = \alpha_{j,i}$ which holds in all three cases.

For $t = 0$, from Equation (3.1) we get

$$\begin{aligned} y_{i,j}(0) &= (\alpha_{i,j}/s_i) \cdot s_i \cdot \ell \\ &= \alpha_{j,i} \cdot \ell \\ &= (\alpha_{j,i}/s_j) \cdot s_j \cdot \ell \\ &= y_{j,i}(0). \end{aligned}$$

Therefore, the load vector remains unchanged and we have $x(1) = x(0)$.

Now suppose that for all $1 \leqslant \tau \leqslant t-1$, we have $x(\tau+1) = x(0)$ and $y_{i,j}(\tau) = y_{j,i}(\tau)$ for all neighbours i, j. We prove that this yields $y_{i,j}(t) = y_{j,i}(t)$ for all i, j, from which it follows that $x(t+1) = x(0)$. From Equation (3.2), for arbitrary neighbours i, j we get:

$$\begin{aligned} y_{i,j}(t) &= (\beta - 1) \cdot y_{i,j}(t-1) + \beta \cdot P_{i,j}(t) \cdot x_i(t) \\ &= (\beta - 1) \cdot y_{j,i}(t-1) + \beta \cdot (\alpha_{i,j}/s_i) \cdot s_i \cdot \ell \\ &= (\beta - 1) \cdot y_{j,i}(t-1) + \beta \cdot (\alpha_{j,i}/s_j) \cdot s_j \cdot \ell \\ &= y_{j,i}(t). \end{aligned}$$

Hence $x(t+1) = x(t) = x(0)$ and the proof follows. □

The next lemma establishes a fact which we will later need to show that under certain conditions our discrete algorithms do not induce negative load.

Lemma 3.2. *Let $\mathbf{x}'$ be an arbitrary load vector on which $\mathcal{A}$ does not induce negative load. Suppose $x^{\mathcal{A}}(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = \ell \cdot (s_1, \ldots, s_n)$ for some $\ell \geqslant 0$. Then for $i \in V$, $t \geqslant 0$, and any $L \subseteq N(i)$ we have:*

$$x_i^{\mathcal{A}}(t) - \sum_{j \in L} \left(y_{i,j}^{\mathcal{A}}(t) - y_{j,i}^{\mathcal{A}}(t) \right) \geqslant s_i \cdot \ell.$$

Proof. By additivity, we have $x_i^{\mathcal{A}}(t) = x_i'(t) + x_i''(t)$ and $y_{i,j}^{\mathcal{A}}(t) = y_{i,j}'(t) + y_{i,j}''(t)$. Thus,

$$x_i^{\mathcal{A}}(t) - \sum_{j \in L} \left(y_{i,j}^{\mathcal{A}}(t) - y_{j,i}^{\mathcal{A}}(t) \right) = x_i'(t) - \sum_{j \in L} \left(y_{i,j}'(t) - y_{j,i}'(t) \right) + x_i''(t) - \sum_{j \in L} \left(y_{i,j}''(t) - y_{j,i}''(t) \right)$$

$$
\begin{aligned}
&= x'_i(t) - \sum_{j \in L} \left(y'_{i,j}(t) - y'_{j,i}(t)\right) + x''_i(t) \qquad (A \text{ is terminating}) \\
&= x'_i(t) - \sum_{j \in L} \left(y'_{i,j}(t) - y'_{j,i}(t)\right) + s_i \cdot \ell \\
&\geqslant x'_i(t) - \sum_{j \in L} y'_{i,j}(t) + s_i \cdot \ell \\
&\geqslant x'_i(t) - \sum_{j \in N(i)} y'_{i,j}(t) + s_i \cdot \ell \\
&\geqslant s_i \cdot \ell,
\end{aligned}
$$

where the last equation follows from the fact that $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$. □

3.3 Deterministic Flow Imitation

In this section, we present and analyze an algorithm that transforms a continuous process $\mathcal{A}$ into its discrete counterpart which we call $\mathfrak{D}(A)$. To distinguish between the two processes we will use $\mathcal{A}$ and $\mathfrak{D}(A)$ as superscripts in the definitions of Section 1.1.

The algorithm (see Algorithm 3.1) keeps track of the total flow $f^{\mathcal{A}}_{i,j}(t)$ that is sent over the edge (i, j) by the continuous algorithm. It calculates the difference in the flow forwarded over the edge by the continuous and the discrete algorithm which we define as

$$\widehat{y}^{\mathcal{A}}_{i,j}(t) := f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1).$$

It then *tries to* find a set $\mathcal{S}_{ij}$ of tasks with a total weight $|\mathcal{S}_{ij}|$ of that difference. These tokens will be forwarded over the edge (i, j). In the case of identical tasks, the amount of load sent from i to its neighbour j is

$$y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) = \left\lfloor f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) \right\rfloor.$$

In the general case of weighted tasks, $\mathcal{S}_{ij}$ is chosen in a way that $f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) \leqslant w_{\max}$.

It might happen that the node i does not contain enough load. In that case the algorithm will create new, artificial tasks and send them to the corresponding neighbours (or equivalently, we may think of an attached infinite source of tokens from which the node gets some tokens). Later we will show that this never happens if the initial load of the resources is large enough.

The following theorem states our result about the transformation introduced by Algorithm 3.1.

Theorem 3.3. *Suppose the discrete process $\mathfrak{D}(A)$ is obtained by transforming a continuous process $\mathcal{A}$ using Algorithm 3.1. Then the following hold:*

(1) If $\mathcal{A}$ does not induce negative load on $x^{\mathcal{A}}(0)$, then for all $t \geqslant T^A$, the max-avg discrepancy of $x^{\mathfrak{D}(\mathcal{A})}(t)$ is at most $2d \cdot w_{\max} + 2$.

(2) If $x^{\mathcal{A}}(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = d \cdot w_{\max} \cdot (s_1, \ldots, s_n)$, and $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$, then the max-min discrepancy of $x^{\mathfrak{D}(\mathcal{A})}(t)$ is at most $2d \cdot w_{\max} + 1$.

Note that among the algorithms mentioned in Section 2.1, only SOS may induce negative load. For other algorithms, the result in part (1) of the above theorem automatically holds, and the condition in part (2) can be translated as having sufficient initial load.

For the special case of identical tasks, the above theorem gives a bound of $2d + 1$ on the max-avg discrepancy. Compared to the deterministic algorithm of [41] that has been analyzed only for hypercubes and tori, we obtain improved bounds for hypercubes while our analysis is more general with respect to network topology, and heterogeneity of tasks and resources. Besides, it also works for the matching-based models and the second order diffusion. See Tables 3.1 and 3.2 for a more detailed comparison of discrepancy bounds.

Before proving the theorem, we need to establish some preliminary results.

Observation 3.4. *As long as Algorithm 3.1 is allowed to access the infinite source we have $|e_{i,j}(t)| < w_{\max}$ for every $(i,j) \in E$ and $t \geqslant 0$.*

Proof. Recall that $e_{i,j}(t) = f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t)$. We observe that $e_{i,j}(t) = -e_{j,i}(t)$. Thus, it suffices to prove the inequality holds for an arbitrary edge direction. In the following, we prove that $f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) < w_{\max}$.

Fix an edge (i,j) and observe that

$$f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) = (-1) \cdot \left(f^{\mathcal{A}}_{j,i}(t) - f^{\mathfrak{D}(\mathcal{A})}_{j,i}(t-1) \right).$$

Assume $f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) \geqslant 0$ (otherwise we switch i and j). From the definition of Algorithm 3.1, it follows that $y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t) = 0$ since $f^{\mathcal{A}}_{j,i}(t) - f^{\mathfrak{D}(\mathcal{A})}_{j,i}(t-1) \leqslant 0$. Therefore,

$$f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) \;=\; f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) + y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t).$$

Algorithm 3.1 $\mathfrak{D}(A)$: Discretized $\mathcal{A}$ using *flow imitation*: the process on node i at round t

for each neighbour j of i
  Compute $f_{i,j}^{\mathcal{A}}(t)$
  $\widehat{y}_{i,j}^{\mathcal{A}}(t) \leftarrow f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t-1)$
  while $\widehat{y}_{i,j}^{\mathcal{A}}(t) - |\mathcal{S}_{ij}| \geqslant w_{\max}$
    if i has no unallocated tasks **then**
      $q \leftarrow$ a unit weight task generated by the attached infinite source
    else
      $q \leftarrow$ arbitrary task removed from the set of unallocated tasks of i
    Add q to $\mathcal{S}_{ij}$
  $y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \leftarrow |\mathcal{S}_{ij}|$

After exiting the loop for node i, we have:

$$f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t-1) - \|\int_{ij}\| < w_{\max},$$

or equivalently,

$$f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t-1) - y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) < w_{\max},$$

which yields:

$$f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) < w_{\max}. \tag{3.5}$$

Let w be the weight of the last task added to $\mathcal{S}_{ij}$. Before adding this task, the loop condition was fulfilled. Hence we have:

$$f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) + w \geqslant w_{\max},$$

and thus:

$$f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \geqslant w_{\max} - w \geqslant 0. \tag{3.6}$$

Combining Equations (3.5) and (3.6), we get

$$\left| f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \right| < w_{\max},$$

as needed. □

Observation 3.5. *For every $(i,j) \in E$ and $t \geqslant 0$ the following holds:*
If $y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) > 0$, then we have

$$y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \leqslant y_{i,j}^{\mathcal{A}}(t) - y_{j,i}^{\mathcal{A}}(t) + e_{i,j}(t-1).$$

Proof. First observe that:

$$\begin{aligned} y^{\mathcal{A}}_{i,j}(t) - y^{\mathcal{A}}_{j,i}(t) + e_{i,j}(t-1) &= y^{\mathcal{A}}_{i,j}(t) - y^{\mathcal{A}}_{j,i}(t) + f^{\mathcal{A}}_{i,j}(t-1) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) \\ &= f^{\mathcal{A}}_{i,j}(t) - f^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) \\ &= \widehat{y}^{\mathcal{A}}_{i,j}(t). \end{aligned}$$

It remains to prove $y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) \leqslant \widehat{y}^{\mathcal{A}}_{i,j}(t)$. Let w be the weight of the last task added to $\mathcal{S}_{ij}$ in round t. Before adding this task, the loop condition of Algorithm 3.1was fulfilled and we had

$$\widehat{y}^{\mathcal{A}}_{i,j}(t) - (|\mathcal{S}_{ij}| - w) \;\geqslant\; w_{\max}.$$

After the loop, we have:

$$y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) = |\mathcal{S}_{ij}|.$$

Consequently,

$$\widehat{y}^{\mathcal{A}}_{i,j}(t) - y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) \geqslant w_{\max} - w \geqslant 0.$$

□

The next lemma is used to prove the discrepancy bound assuming no token is borrowed from the infinite source.

Lemma 3.6. *For $i \in V$ and $\tau \geqslant 0$, suppose i does not use its infinite source by the end of the round $\tau - 1$. Then for any $t \leqslant \tau$ the following hold:*

(1) $x^{\mathfrak{D}(\mathcal{A})}_i(t) = x^{\mathcal{A}}_i(t) + \sum_{j \in N(i)} e_{i,j}(t-1)$.

(2) $\left| x^{\mathfrak{D}(\mathcal{A})}_i(t) - x^{\mathcal{A}}_i(t) \right| < d \cdot w_{\max}$.

Proof. The proof of (1) is by induction on t. For $t = 0$, we have $x^{\mathfrak{D}(\mathcal{A})}_i(0) = x^{\mathcal{A}}_i(0)$ and for all i, j, $E_{i,j}(-1) = 0$. Therefore the equation holds.

Suppose for some $t \geqslant 0$ we have

$$x^{\mathfrak{D}(\mathcal{A})}_i(t) = x^{\mathcal{A}}_i(t) + \sum_{j \in N(i)} e_{i,j}(t-1).$$

It remains to prove that the statement holds for $t+1$ as well. As long as $t+1 \leqslant \tau$, we can apply Equation (1.1) to get:

$$x^{\mathfrak{D}(\mathcal{A})}_i(t+1) = x^{\mathfrak{D}(\mathcal{A})}_i(t) + \sum_{j \in N(i)} (y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t) - y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t))$$

$$
\begin{aligned}
&= x_i^{\mathcal{A}}(t) + \sum_{j \in N(i)} e_{i,j}(t-1) + \sum_{j \in N(i)} \left(y_{j,i}^{\mathfrak{D}(\mathcal{A})}(t) - y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \right) \\
&= x_i^{\mathcal{A}}(t) + \sum_{j \in N(i)} \left(f_{i,j}^{\mathcal{A}}(t-1) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t-1) + y_{j,i}^{\mathfrak{D}(\mathcal{A})}(t) - y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \right) \\
&= \left(x_i^{\mathcal{A}}(t+1) - \sum_{j \in N(i)} (y_{j,i}^{\mathcal{A}}(t) - y_{i,j}^{\mathcal{A}}(t)) \right) + \sum_{j \in N(i)} \left(f_{i,j}^{\mathcal{A}}(t-1) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \right) \\
&= x_i^{\mathcal{A}}(t+1) + \sum_{j \in N(i)} \left(f_{i,j}^{\mathcal{A}}(t-1) + y_{i,j}^{\mathcal{A}}(t) - y_{j,i}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \right) \\
&= x_i^{\mathcal{A}}(t+1) + \sum_{j \in N(i)} \left(f_{i,j}^{\mathcal{A}}(t) - f_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \right) \\
&= x_i^{\mathcal{A}}(t+1) + \sum_{j \in N(i)} e_{i,j}(t).
\end{aligned}
$$

This finishes the proof of (1). For (2), we apply (1) to get

$$
\left| x_i^{\mathfrak{D}(\mathcal{A})}(t) - x_i^{\mathcal{A}}(t) \right| = \left| \sum_{j \in N(i)} e_{i,j}(t-1) \right| \leqslant \sum_{j \in N(i)} |e_{i,j}(t-1)|.
$$

Now, the result can be obtained using Observation 3.4. □

The following lemma shows that if there is initially sufficient amount of load on each node, then $D(A)$ does not induce negative load on $x^{\mathfrak{D}(\mathcal{A})}(t)$.

Lemma 3.7. *Suppose $x(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = d \cdot w_{\max} \cdot (s_1, \ldots, s_n)$, and $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$.*

Then for all $i \in V$ and $t \geqslant 0$, the following holds:

$$
x_i^{\mathfrak{D}(\mathcal{A})}(t) - \sum_{j \in N(i)} y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \geqslant 0.
$$

Proof. For the sake of contradiction, let us assume there is some round t_1 in which we use the infinite source for the first time. Let i be an arbitrary node with insufficient load, so that we have $x_i^{\mathfrak{D}(\mathcal{A})}(t_1) - \sum_{j \in N(i)} y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t_1) < 0$. Define

$$
L := \{ j \in N(i) : y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t_1) > 0 \}
$$

to be the set of neighbours of node i to which i transfers load in the round t_1. We get:

$$
x_i^{\mathfrak{D}(\mathcal{A})}(t_1) - \sum_{j \in N(i)} y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t_1)
$$

$$
\begin{aligned}
&= x_i^{\mathfrak{D}(\mathcal{A})}(t_1) - \sum_{j\in L} y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t_1) \\
&\geqslant x_i^{\mathcal{A}}(t_1) + \sum_{j\in N(i)} e_{i,j}(t_1-1) - \sum_{j\in L}\big(y_{i,j}^{\mathcal{A}}(t_1) - y_{j,i}^{\mathcal{A}}(t_1) + e_{i,j}(t_1-1)\big) && (3.7)\\
&= x_i^{\mathcal{A}}(t_1) - \sum_{j\in L}\big(y_{i,j}^{\mathcal{A}}(t_1) - y_{j,i}^{\mathcal{A}}(t_1)\big) + \sum_{j\in N(i)-L} e_{i,j}(t_1-1) \\
&= d\cdot s_i\cdot w_{\max} + \sum_{j\in N(i)-L} e_{i,j}(t_1-1) && (3.8)\\
&\geqslant d\cdot s_i\cdot w_{\max} - |N(i)-L|\cdot w_{\max} && (3.9)\\
&\geqslant d\cdot s_i\cdot w_{\max} - d\cdot w_{\max} \geqslant 0, \qquad\qquad (\text{since } s_i \geqslant 1)
\end{aligned}
$$

where in Equation (3.7) we use Observation 3.5 and Lemma 3.6, and the fact that no infinite source has been used before the round t_1. Equation (3.8) follows from the Lemma 3.2 using $x(0) = \mathbf{x}' + \mathbf{x}''$ and the conditions on $\mathbf{x}'$ and $\mathbf{x}''$ mentioned in the statement of the lemma, and by setting $\ell = d\cdot w_{\max}$. Also, Equation (3.9) results from Observation 3.4.

This contradicts our initial assumption that

$$x_i^{\mathfrak{D}(\mathcal{A})}(t_1) - \sum_{j\in N(i)} y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t_1) < 0,$$

and the proof follows. □

We are now ready to prove Theorem 3.3.

Proof. First we prove part (2). Suppose $x^{\mathcal{A}}(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = d\cdot w_{\max}\cdot (s_1,\ldots,s_n)$, and $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$. Then by Lemma 3.7 we know that negative load never occurs. Therefore, no infinite source is used and the total load remains intact. Hence, by part (2) of the Lemma 3.6, we have:

$$\left|x_i^{\mathfrak{D}(\mathcal{A})}(t) - x_i^{\mathcal{A}}(t)\right| < d\cdot w_{\max}.$$

This fact together with the fact that after the balancing time we have

$$\left|x_i^{\mathcal{A}}(t) - w\cdot s_i/s\right| \leqslant 1,$$

yield

$$\left|x_i^{\mathfrak{D}(\mathcal{A})}(t) - w\cdot s_i/s\right| < d\cdot w_{\max} + 1.$$

Since $s_i \geqslant 1$, we have $\left|x_i^{\mathfrak{D}(\mathcal{A})}(t)/s_i - w/s\right| < d \cdot w_{\max} + 1$, which holds for arbitrary node i. Hence, for any pair of nodes i, j we get $\left|x_i^{\mathfrak{D}(\mathcal{A})}(t)/s_i - x_j^{\mathfrak{D}(\mathcal{A})}(t)/s_j\right| < 2d \cdot w_{\max} + 2$ which gives the desired max-min discrepancy bound.

In the general case, to get the bound of part (1) the algorithm first adds $d \cdot s_i \cdot w_{\max}$ *dummy* unit weight tasks to each resource i before the process begins. Note that this does not affect the convergence time of the continuous process, because the extra load is completely balanced. In the rest of the proof, we use x to refer to the new load vectors. Let w' and w denote the original and the new total load, respectively. We have:

$$\begin{aligned} w &= w' + \sum_i d \cdot s_i \cdot w_{\max} \\ &= w' + d \cdot s \cdot w_{\max}. \end{aligned}$$

Hence,

$$w/s \leqslant w'/s + d \cdot w_{\max}$$

At the end, the dummy tokens can be simply ignored. Nevertheless we can still use $x_i^{\mathfrak{D}(\mathcal{A})}(t)$ as an upper bound on the final load of the node i excluding the dummy tokens. Following steps similar to the max-min discrepancy case, we get

$$x_i^{\mathfrak{D}(\mathcal{A})}(t)/s_i - w/s < d \cdot w_{\max} + 1,$$

as required. □

3.4 Randomized Flow Imitation

In this section we analyze a randomized version of Algorithm 3.1 that can be applied for balancing identical tasks. Instead of always rounding down the flow that has to be sent over an edge, Algorithm 3.2 uses randomized rounding. The notation we use in this section is the same notation used in Section 3.3, except here we use uppercase letters to express random variables. As an example $f_{i,j}^{\mathcal{A}}(t)$ is the flow sent over edge (i, j) in round t by the continuous process, while $F_{i,j}^{\mathfrak{D}(\mathcal{A})}(t)$ is the corresponding random variable for the discrete process.

Algorithm 3.2 calculates the flow

$$\widehat{Y}_{i,j}(t) := f_{i,j}^{\mathcal{A}}(t) - F_{i,j}^{\mathfrak{D}(\mathcal{A})}(t-1)$$

that has to be sent over edge (i, j) as before. To calculate the flow that is actually sent, $\widehat{Y}_{i,j}(t)$ is randomly rounded up or down, with a probability depending on the value of its fractional part. Suppose $\widehat{Y}_{i,j}(t) > 0$ in some round t. Then the discrete flow $Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t)$ that is sent over the edge is a random variable determined by the following randomized rounding scheme. So we have

$$Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) = \begin{cases} \lfloor \widehat{Y}_{i,j}(t) \rfloor + 1 & \text{with probability } \{\widehat{Y}_{i,j}(t)\}, \\ \lfloor \widehat{Y}_{i,j}(t) \rfloor & \text{otherwise.} \end{cases}$$

In Algorithm 3.2 we use $Z_{i,j}(t)$, which is a zero-one random variable indicating whether we should round up. Once we know all the random choices in round t, we can calculate the load of processor i as before using

$$X_i^{\mathfrak{D}(\mathcal{A})}(t+1) = X_i^{\mathfrak{D}(\mathcal{A})}(t) - \sum_{j \in N(i)} (Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) - Y_{j,i}^{\mathfrak{D}(\mathcal{A})}(t)).$$

Algorithm 3.2 $\mathfrak{D}$(A): Discretized A using randomized *flow imitation*: the process on node i at round t

for each neighbour j of i in parallel
  Compute $f_{i,j}^{\mathcal{A}}(t)$
  $\widehat{Y}_{i,j}(t) \leftarrow f_{i,j}^{\mathcal{A}}(t) - F_{i,j}^{\mathfrak{D}(\mathcal{A})}(t-1)$
  if $\widehat{Y}_{i,j}(t) > 0$ **then**
    Toss a coin with head probability $\{\widehat{Y}_{i,j}(t)\}$
    $Z_{i,j}(t) \leftarrow \begin{cases} 1 & \text{if head comes up;} \\ 0 & \text{otherwise.} \end{cases}$
    $Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \leftarrow \lfloor \widehat{Y}_{i,j}(t) \rfloor + Z_{i,j}(t)$
    Send $Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t)$ tokens to j
    if there are not enough tokens **then**
      generate the required amount using the attached infinite source

We will show that with high probability the roundings errors sum up to a small value. The following theorem states our result about the transformation introduced by Algorithm 3.2.

Theorem 3.8. *Suppose the discrete process $\mathfrak{D}(A)$ is obtained by transforming a continuous process $\mathcal{A}$ using Algorithm 3.2 and that $T^A \leqslant n^\kappa$ for some constant κ. Then the following hold:*

(1) If $\mathcal{A}$ does not induce negative load on $x^{\mathcal{A}}(0)$, then the max-avg discrepancy of $X^{\mathfrak{D}(\mathcal{A})}(T^A)$ is at most $d/4 + \mathcal{O}(\sqrt{d \log n})$ w.h.p.

(2) If $x^{\mathcal{A}}(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = (d/4 + 2c \cdot \sqrt{d \log n}) \cdot (s_1, \ldots, s_n)$ for a properly chosen constant $c > 0$ and $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$, then w.h.p. the max-min discrepancy of $X^{\mathfrak{D}(\mathcal{A})}(T^A)$ is $\mathcal{O}(\sqrt{d \log n})$.

Note that among the algorithms mentioned in Section 2.1, only SOS may induce negative load. For other algorithms, the result in part (1) of the above theorem automatically holds, and the condition in part (2) can be translated as having sufficient initial load.

The next observation provides useful tools for proving the above theorem.

Lemma 3.9. *For $i \in V$, $j \in N(i)$ and $t \geqslant 0$ the following hold:*

(1) $E_{i,j}(t) = y^{\mathcal{A}}_{i,j}(t) - y^{\mathcal{A}}_{j,i}(t) + E_{i,j}(t-1) - (Y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) - Y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t))$.

(2) If $Y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) > 0$ then $Y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t) = 0$.

(3) If $\widehat{Y}_{i,j}(t) > 0$, then we have[5]:

$$E_{i,j}(t) = \begin{cases} \{\widehat{Y}_{i,j}(t)\} - 1 & \text{if } Z_{i,j}(t) = 1 \\ \{\widehat{Y}_{i,j}(t)\} & \text{otherwise.} \end{cases}$$

Proof. To prove the first statement, recall that $E_{i,j}(t) = f^{\mathcal{A}}_{i,j}(t) - F^{\mathfrak{D}(\mathcal{A})}_{i,j}(t)$. The right side of the equation can be simplified as below:

$$\begin{aligned} & y^{\mathcal{A}}_{i,j}(t) - y^{\mathcal{A}}_{j,i}(t) + E_{i,j}(t-1) - \left(Y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) - Y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t)\right) \\ &= \left(y^{\mathcal{A}}_{i,j}(t) - y^{\mathcal{A}}_{j,i}(t) + f^{\mathcal{A}}_{i,j}(t-1)\right) - \left(F^{\mathfrak{D}(\mathcal{A})}_{i,j}(t-1) + Y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) - Y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t)\right) \\ &= f^{\mathcal{A}}_{i,j}(t) - F^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) \\ &= E_{i,j}(t). \end{aligned}$$

For the second statement, note that if $Y^{\mathfrak{D}(\mathcal{A})}_{i,j}(t) > 0$ then according to Algorithm 3.2, $\widehat{Y}_{i,j}(t) > 0$ must be satisfied; therefore $f^{\mathcal{A}}_{j,i}(t) - F^{\mathfrak{D}(\mathcal{A})}_{j,i}(t-1) < 0$ which yields $Y^{\mathfrak{D}(\mathcal{A})}_{j,i}(t) = 0$.

[5] *Recall that for a real number a, $\{a\} := a - \lfloor a \rfloor$*

Now we prove the third statement. Since $\widehat{Y}_{i,j}(t) > 0$, we have $\widehat{Y}_{j,i}(t) = -\widehat{Y}_{i,j}(t) < 0$ and therefore $Y_{j,i}^{\mathfrak{D}(\mathcal{A})}(t) = 0$ by part (2) of the observation. Thus, using part (1) we get

$$\begin{aligned} E_{i,j}(t) &= y_{i,j}^{\mathcal{A}}(t) - y_{j,i}^{\mathcal{A}}(t) + E_{i,j}(t-1) - Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \\ &= \widehat{Y}_{j,i}(t) - Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \\ &= \widehat{Y}_{j,i}(t) - \lfloor \widehat{Y}_{i,j}(t) \rfloor - Z_{i,j}(t), \end{aligned}$$

where the last equation follows from the way $Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t)$ is obtained in Algorithm 3.2, and the proof follows from the fact that $Z_{i,j}(t)$ can be either zero or one. □

Note that part (3) of the Lemma 3.9 shows that $\mathbf{Ex}\left[e_{i,j}(t)\right] = 0$.

Now we show that w.h.p. the discrete process does not deviate much from the continuous process. We identify some *undesirable* events as listed in the Lemma 3.10 and show in that each of these events happens only with a small probability. In our proofs, we make use of Lemma A.3 in the appendix which is a simple adaptation of the Hoeffding's bound [47] for sums of randomized rounding errors.

Before stating the lemma, we introduce some notation. Define

$$H_i(t) := \{j \in N(i) : y_{i,j}^{\mathcal{A}}(t) - y_{j,i}^{\mathcal{A}}(t) + E_{i,j}(t-1) > 0\}$$

as the set of neighbours of i to which i may send tokens in round t. Let

$$L_i(t) := N(i) - H_i(t)$$

be the rest of i's neighbours.

Lemma 3.10. *Suppose there is a constant $\kappa > 0$ so that $T^A \leqslant n^\kappa$. Then for any constant $\alpha > 0$ there is a constant $c(\kappa, \alpha) > 0$[6] such that for any node i and round $t \leqslant T^A$, each of the following events occurs with probability at most $(n^{\alpha+1} \cdot T^A)^{-1}$:*

(1) $\left|\sum_{j \in N(i)} E_{i,j}(t)\right| \geqslant c \cdot \sqrt{d \log n}$,

(2) $\left|\sum_{j \in H_i(t)} E_{i,j}(t)\right| \geqslant c \cdot \sqrt{d \log n}$,

(3) $\sum_{j \in L_i(t+1)} E_{i,j}(t) \leqslant -\frac{d}{4} - c \cdot \sqrt{d \log n}$.

[6] *It is sufficient to choose $c(\kappa, \alpha) \approx \sqrt{\kappa + \alpha}$.*

Proof. First we prove statement (1). Define $\Delta := \sum_{j \in N(i)} E_{i,j}(t)$.

Assume $E_{i,j}(t-1)$ is fixed for all the edges (i,j). Then each of the random variables $E_{i,j}(t)$ can assume at most two different values and rounding up or down is independent of other edges (see part (3) of Lemma 3.9). Let the random variable $\mathbf{e}_i(t)$ be a vector denoting the error values of the edges connected to i at the end of the round t. By the law of total probability we have:

$$\mathbf{Pr}\left[\ |\Delta| \geqslant \delta\ \right] = \sum_{\mathcal{E}_i} \mathbf{Pr}\left[\ |\Delta| \geqslant \delta \mid \mathbf{e}_i(t-1) = \mathcal{E}_i\ \right] \cdot \mathbf{Pr}\left[\ \mathbf{e}_i(t-1) = \mathcal{E}_i\ \right].$$

Note that each $E_{i,j}(t)$ is the random variable indicating the error in the randomized rounding of $\widehat{Y}_{i,j}(t)$ (part (3) of Lemma 3.9). We can apply Lemma A.3 in the appendix to bound Δ, which yields

$$\mathbf{Pr}\left[|\Delta| \geqslant \delta \mid \mathbf{e}_i(t-1) = \mathcal{E}_i\right] \leqslant 2\exp\left(-2\delta^2/d\right).$$

Hence,

$$\begin{aligned}
\mathbf{Pr}\left[\ |\Delta| \geqslant \delta\ \right] &= \sum_{\mathcal{E}_i} \mathbf{Pr}\left[\ |\Delta| \geqslant \delta \mid \mathbf{e}_i(t-1) = \mathcal{E}_i\ \right] \cdot \mathbf{Pr}\left[\ \mathbf{e}_i(t-1) = \mathcal{E}_i\ \right] \\
&\leqslant \sum_{\mathcal{E}_i} 2\exp\left(-2\delta^2/d\right) \cdot \mathbf{Pr}\left[\ \mathbf{e}_i(t-1) = \mathcal{E}_i\ \right] \\
&= 2\exp\left(-2\delta^2/d\right).
\end{aligned}$$

As $T^A \leqslant n^\kappa$, setting $\delta = c \cdot \sqrt{d \log n} \geqslant \sqrt{d \log\left(2n^{\alpha+1}T^A\right)/2}$ for some constant c yields the desired bound.

The proof of statement (2) is similar to the proof of (1). Here we define

$$\Delta := \sum_{j \in H_i(t)} E_{i,j}(t).$$

Recall the definition of $H_i(t) = \{j \in N(i) : y^{\mathcal{A}}_{i,j}(t) - y^{\mathcal{A}}_{j,i}(t) + E_{i,j}(t-1) > 0\}$. Observe that $|H_i(t)| \leqslant d$. Conditioned on $\mathbf{e}_i(t-1) = \mathcal{E}_i$, the set $H_i(t)$ is fixed. Hence we can apply Lemma A.3 to obtain:

$$\mathbf{Pr}\left[|\Delta| \geqslant \delta \mid \mathbf{e}_i(t-1) = \mathcal{E}_i\right] \leqslant 2\exp\left(-2\delta^2/d\right).$$

Following the same steps as in (1) we get the desired result.

Now we proceed with the proof of statement (3). Recall that

$$L_i(t+1) = \{j \in N(i) : y^{\mathcal{A}}_{i,j}(t+1) - y^{\mathcal{A}}_{j,i}(t+1) + E_{i,j}(t) \leqslant 0\},$$

so intuitively the set is biased toward containing lower values of $E_{i,j}(t)$. However, we can change the summation and use different random variables so that we can still apply Hoeffding's bounds. Define $E^-_{i,j}(t) := \min\{E_{i,j}(t), 0\}$. We have:

$$\sum_{j \in N(i)} E^-_{i,j}(t) \leqslant \sum_{j \in L_i(t+1)} E_{i,j}(t). \tag{3.10}$$

Fix an arbitrary node i, let $\Delta = \sum_{j \in N(i)} E^-_{i,j}(t)$ and $p_{ij} = \{\widehat{Y}_{i,j}(t)\}$. We have:

$$\mathbf{Ex}\left[E^-_{i,j}(t) \;\middle|\; \mathbf{e}_i(t-1) = \mathcal{E}_i\right] = -p_{ij} \cdot (1 - p_{ij}) + 0 \cdot p_{ij} \geqslant -1/4.$$

The last step follows from a simple minimization of $f(p_{ij}) = (1-p_{ij})+0\cdot p_{ij}$. Conditioned on a fixed $\mathbf{e}_i(t-1)$, the random variables $E^-_{i,j}(t)$ are independent ranging over intervals of length no more than one. Again, we can apply Hoeffding's bounds using $\delta = c\cdot\sqrt{d \log n} \geqslant \sqrt{d \log (2n^{\alpha+1}T^A)/2}$ with the same constant c as in the previous parts. Hence,

$$\mathbf{Pr}\left[\Delta < -\frac{d}{4} - c \cdot \sqrt{d \log n} \;\middle|\; \mathbf{e}_i(t-1) = \mathcal{E}_i\right] \leqslant \frac{1}{2n^{\alpha+1} \cdot T^A}. \tag{3.11}$$

By Equation (3.10), we can replace Δ with $\sum_{j \in L_i(t+1)} E_{i,j}(t)$ in the Equation (3.11) to obtain the desired bound. □

The next lemma provides the two main ingredients for proving the Theorem 3.8. First, the discrete process stays close to the continuous process; and second, that this happens without accessing the infinite sources.

Lemma 3.11. *Suppose there is a constant $\kappa > 0$ so that $T^A \leqslant n^\kappa$. Further suppose $x^{\mathcal{A}}(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = (d/4 + 2c \cdot \sqrt{d \log n}) \cdot (s_1, \ldots, s_n)$, and $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$. Then for any constant $\alpha > 0$ there is a constant $c(\kappa, \alpha) > 0$ such that the following holds:*

(1) $\mathbf{Pr}\Big[\ \Big|\sum_{j \in N(i)} E_{i,j}(t)\Big| \leqslant c \cdot \sqrt{d \log n} \quad \text{holds for all } i \in V \quad \text{and } t \leqslant T^A\ \Big] \geqslant 1 - n^{-\alpha}$,

(2) No infinite source is used, i.e.,
$\mathbf{Pr}\Big[\ X_i^{\mathfrak{D}(\mathcal{A})}(t) - \sum_{j \in H_i(t)} Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t) \ \geqslant\ 0 \quad \text{holds for all } t \leqslant T^A \text{ and } i \in V\ \Big] \geqslant 1 - 2n^{-\alpha}$.

Proof. To prove statement (1), we choose the value of c as computed by Lemma 3.10 for the same value of α and $t_1 = T^A$. The proof of the first statement follows by applying the union bound to part (1) of Lemma 3.10.

To prove statement (2), we need to show that the load of every node i at the beginning of every round $t \leqslant T^A$ is large enough to satisfy their outgoing demands. We prove by contradiction, assuming there is a first round $t' \leqslant T^A$ in which some node i has insufficient load. Recall that $H_i(t)$ is defined so that no load is transferred from i to any of its neighbours not in $H_i(t)$. In the following we prove that $X_i^{\mathfrak{D}(\mathcal{A})}(t') - \sum_{j \in H_i(t')} Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t') \geqslant 0$, contradicting the initial assumption that i does not have sufficient load in round t'.

$$
\begin{aligned}
& X_i^{\mathfrak{D}(\mathcal{A})}(t') - \sum_{j \in H_i(t')} Y_{i,j}^{\mathfrak{D}(\mathcal{A})}(t') \\
&= X_i^{\mathfrak{D}(\mathcal{A})}(t') - \sum_{j \in H_i(t')} \left(y_{i,j}^{\mathcal{A}}(t') - y_{j,i}^{\mathcal{A}}(t') + E_{i,j}(t'-1) - E_{i,j}(t') \right) && (3.12) \\
&= x_i^{\mathcal{A}}(t') + \sum_{j \in N(i)} E_{i,j}(t'-1) - \sum_{j \in H_i(t')} \left(y_{i,j}^{\mathcal{A}}(t') - y_{j,i}^{\mathcal{A}}(t') + E_{i,j}(t'-1) - E_{i,j}(t') \right) && (3.13) \\
&= x_i^{\mathcal{A}}(t') - \sum_{j \in H_i(t')} \left(y_{i,j}^{\mathcal{A}}(t') - y_{j,i}^{\mathcal{A}}(t') \right) + \sum_{j \in L_i(t')} E_{i,j}(t'-1) + \sum_{j \in H_i(t')} E_{i,j}(t') \\
&\geqslant s_i \cdot (d/4 + 2c \cdot \sqrt{d \log n}) + \sum_{j \in L_i(t')} E_{i,j}(t'-1) + \sum_{j \in H_i(t')} E_{i,j}(t'), && (3.14)
\end{aligned}
$$

where in Equation (3.12) we use parts (1) and (2) of the Lemma 3.9, Equation (3.13) follows from the part (1) of the Lemma 3.6 using the fact that no infinite source is used before the round t'. Also, Equation (3.14) is obtained by applying Lemma 3.2 using $\ell = (d/4 + 2c \cdot \sqrt{d \log n})$.

To complete the proof, it suffices to consider $s_i \geqslant 1$ and apply the parts (2) and (3) of the Lemma 3.10 to the above equation using the union bound. □

We are now ready to prove the Theorem 3.8.

Proof. First we prove part (2). Suppose $x^{\mathcal{A}}(0) = \mathbf{x}' + \mathbf{x}''$ such that $\mathbf{x}'' = (d/4 + 2c \cdot \sqrt{d \log n}) \cdot (s_1, \ldots, s_n)$, and $\mathcal{A}$ does not induce negative load on $\mathbf{x}'$. Consider an arbitrary constant $\alpha > 0$, and let $c = 2c'$ where c' is the constant computed in Lemma 3.11 using the same α. Applying the union bound to combine both parts of the Lemma 3.11 we get with probability of at least $1 - 3n^{-\alpha}$ that no infinite source is ever used and

that $|\sum_{j\in N(i)} E_{i,j}(t)| < c' \cdot \sqrt{d \log n}$. As no infinite source is used, by part (1) of the Lemma 3.6 [7] we also get $|X_i^{\mathfrak{D}(\mathcal{A})}(t) - x_i^{\mathcal{A}}(t)| < c' \cdot \sqrt{d \log n}$.

On the other hand, using the definition of the balancing time we get $|x_i^{\mathcal{A}}(t) - w \cdot s_i/s| \leqslant 1$. Hence, we can conclude that

$$\left|X_i^{\mathfrak{D}(\mathcal{A})}(t) - w \cdot s_i/s\right| < c' \cdot \sqrt{d \log n} + 1.$$

Since $s_i \geqslant 1$, we have $|X_i^{\mathfrak{D}(\mathcal{A})}(t)/s_i - w/s| < c' \cdot \sqrt{d \log n} + 1$ which holds for every node i. Hence, for any pair of nodes i, j we get

$$\left|X_i^{\mathfrak{D}(\mathcal{A})}(t)/s_i - X_j^{\mathfrak{D}(\mathcal{A})}(t)/s_j\right| < 2c' \cdot \sqrt{d \log n} + 2,$$

yielding the desired max-min discrepancy bound.

To get the bound of part (1), the algorithm first adds $(d/4 + 2c \cdot \sqrt{d \log n}) \cdot s_i$ *dummy* unit weight tasks to each resource i before the process begins. Note that this does not affect the convergence time of the continuous process, because the extra load is completely balanced. In the rest of the proof, we use x to refer to the new load vectors. Let w' and w denote the original and the new total load, respectively. We have: $w = w' + (d/4 + 2c \cdot \sqrt{d \log n}) \cdot s$. Hence,

$$w/s \leqslant w'/s + d/4 + 2c \cdot \sqrt{d \log n}.$$

At the end, the dummy tokens can be simply ignored. Though, we can still use $X_i^{\mathfrak{D}(\mathcal{A})}(t)$ as an upper bound on the final load of the node i excluding the dummy tokens. Following steps similar to the max-min discrepancy case, we get $X_i^{\mathfrak{D}(\mathcal{A})}(t)/s_i - w/s < d/4 + \mathcal{O}(\sqrt{d \log n})$, which yields the desired max-avg discrepancy bound. □

3.5 Comparison with Other Results

Tables 3.1 and 3.2 compare our algorithms with previous results. For easier comparison, our results are translated to the case of uniform tasks and speeds. Table 3.1 compares our algorithms with other diffusion algorithms. Algorithm 3.1 achieves a final max-min discrepancy independent of n and graph expansion, and in particular, the only algorithm achieving constant max-min discrepancy for all constant-degree graphs. In

[7] It is easy to see that Lemma 3.6 also holds for the randomized scheme.

the matching model (Table 3.2), Algorithm 3.1 is the only algorithm that achieves final max-min discrepancy independent of n for an arbitrary, possibly non-regular graph.

Algorithm 3.2 improves over the results of [10, 41, 62] by reaching a max-min discrepancy independent of graph expansion for arbitrary graphs (see Table 3.2). Compared to the results of [64] for randomized diffusion, Algorithm 3.2 saves a factor of $\sqrt{d}$ when the loop probability is $1/2$ and a factor of d if the loop probability is $1/(d+1)$. Similarly, in the matching model, Algorithm 3.2 achieves max-min discrepancy bounds independent of graph expansion, thus giving improved bounds compared to [39, 62, 64] for low-expansion graphs. Note that the probability of achieving the bounds is also better than [64].

Compared to the existing results on non-uniform speeds [2, 30], both Algorithm 3.1 and Algorithm 3.2 achieve max-min discrepancy bounds independent of global graph parameters while previous bounds depend on the expansion [2, 30] or the diameter of the graphs [2].

Table 3.1: **Final discrepancy of our algorithms compared to other discrete diffusion processes for different graph classes.** The running time of each process is $T = \mathcal{O}(\frac{\log Kn}{1-\lambda})$.

Discrete Processes	**Arbitrary Graphs**	**Expanders with $d = \mathcal{O}(1)$**	**Hypercubes**	**r-dim tori $r = \mathcal{O}(1)$**
Deterministic Rounding				
Rabani et al. [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/r})$
Friedrich et al. [41] (deterministic)	–	–	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(1)$
Alg. 3.1 (Theorem 3.3)	$\mathcal{O}(d)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
Randomized Rounding				
Friedrich et al. [41] (randomized)	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$	$\mathcal{O}(\log\log n)$	$\mathcal{O}(\log^2 n \log\log n)$	$\mathcal{O}(n^{1/r} \log\log n)$
Berenbrink et al. [10]	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$, and $\mathcal{O}\left(d\sqrt{\log n} + \sqrt{\frac{\log n \log d}{1-\lambda}}\right)$	$\mathcal{O}(\log\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$
Sauerwald and Sun [64] applied to algorithm of [10]	$\mathcal{O}(d\sqrt{d \log n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(\sqrt{\log n})$
Sauerwald and Sun [64] applied to algorithm of [41]	$\mathcal{O}(\sqrt{d \log n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$
Alg. 3.2 (Theorem 3.8)	$\mathcal{O}(\sqrt{d \log n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$

Table 3.2: **Final discrepancy compared to other discrete processes in the matching model.**

Discrete Processes	**Arbitrary Graphs**	**Expanders with $d = \mathcal{O}(1)$**	**Hypercubes**	**r-dim tori $r = \mathcal{O}(1)$**
Periodic Matchings				
Round-Down				
Rabani et al. [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/r})$
Randomized Rounding				
Friedrich and Sauerwald [39]	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$, and $\mathcal{O}\left(\sqrt{\frac{d \log n}{1-\lambda}}\right)$	$\mathcal{O}(\log\log n)$, and $\mathcal{O}(1)^{\dagger}$	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(n^{1/2r}\sqrt{\log n})$
Sauerwald and Sun [64]	$\mathcal{O}(\log^{\varepsilon} n)^{*}$, and $\mathcal{O}(\log\log n)^{\P}$	$\mathcal{O}(1)^{*}$	$\mathcal{O}(\log^{\varepsilon} n)^{*}$	$\mathcal{O}(1)^{*}$
Random Matchings				
Round-Down				
Rabani et al. [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/r})$
Randomized Rounding				
Friedrich and Sauerwald [39]	$\mathcal{O}\left(\sqrt{\frac{\log^3 n}{1-\lambda}}\right)$	$\mathcal{O}(1)^{\dagger}$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/2r}\log n)$
Sauerwald and Sun [64]	$\mathcal{O}(\log^{\varepsilon} n)^{*}$, and $\mathcal{O}(\log\log n)^{\P}$	$\mathcal{O}(1)^{*}$	$\mathcal{O}(1)^{*}$	$\mathcal{O}(1)^{*}$
Periodic/Random Matchings				
Alg. 3.1: Round-Down	$\mathcal{O}(d)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
Alg. 3.2: Randomized Rounding	$\mathcal{O}(\sqrt{d \log n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$

[*] Unlike other probabilistic bounds that hold with probability $1 - n^{-\Omega(1)}$, these bounds hold with probability $1 - \exp(-(\log n)^c)$, for some $c \ll 1$

[†] in $t = \mathcal{O}(T \cdot \log^3 \log n)$ rounds

[¶] in $t = \mathcal{O}(T \cdot \log\log n)$ rounds

Chapter 4

Propp Machines

In this chapter[1] we consider the *rotor walk* model on regular graphs, which works as follows: A d-regular graph is given, along with an arbitrary distribution of tokens over its nodes. The process works in rounds and in every *round* each node distributes all of its tokens to its neighbours. We assume that every node is equipped with a rotor and its neighbours are arranged in a fixed circular list. At the beginning of a round, the rotor is pointing toward the neighbour that got the last token in the previous round. Then the rotor is directed to the next neighbour in the circular list, one token is allocated to that neighbour, and so on, until no token remains. A rotor walk can be regarded as a derandomized version of a random walk. Several publications address the question of how closely rotor walks approximate the expected token distributions of random walk models. The closeness of the two models is usually measured by comparing the number of tokens a node has in the two models, taking the maximum difference between the two values over every node and every round (see Section 2.3.3 for the related work on this model). In this chapter, we analyze the behaviour of rotor walks as diffusion load balancing schemes[2]. To the best of our knowledge, our work is the first to analyze Propp machines in the context of load balancing. Though the idea of distributing tokens in a round-robin fashion was mentioned before [62], it was never analyzed. We consider two rotor walk models, a deterministic model called *D_Propp* and a randomized model

[1] An extended abstract of the material in this chapter has been published in the *proceedings of the twenty-fifth annual ACM symposium on parallelism in algorithms and architectures* (SPAA'13) [4].

[2] This is reason the why we only discuss regular graphs. If the graph is not regular, the rotor walk will not converge to the uniform distribution which is not in line with the goal of load balancing.

called *R_Propp*[3]. We obtain deviation bounds for the rotor walk model. These can be compared to discrepancy bounds in the diffusion model for the following reason: One well-known technique to analyze a discrete process is to bound the deviation between the discrete process and a continuous counterpart which is equivalent to a Markov chain with uniform steady-state distribution. After the convergence time of the continuous process, the deviation of the two processes captures the discrepancy of the discrete process. Hence, in this context one can use deviation bounds and discrepancy bounds (almost) interchangeably.

D_Propp. In Section 4.1 we study a deterministic discrete diffusion algorithm based on the Propp machines. Similar to [41], *D_Propp* follows a *lazy* continuous process where every node keeps half of the tokens and distributes the other half using a rotor. The rotor points to neighbour $i+1$ at the beginning of a round when the rotor stopped at i in the previous round. Our analysis relies on the result by Friedrich et al. [41], which is based on the unimodality of transition probabilities, a property that requires loop probabilities of at least $1/2$.

Our main contribution in analyzing *D_Propp* is proving that it exhibits the *bounded-error property* on any graph (Theorem 4.1). Hence, similar to the process of [41], we get deviation bounds of $\mathcal{O}(\log^{3/2} n)$ for hypercubes and $\mathcal{O}(1)$ for constant-degree tori. Since our analysis is based on the analysis of [41], due to the same technical limitations in that paper we were not able to obtain improved bounds for other graph classes. Compared to the process of [41], our algorithm is simpler and avoids negative loads, thus achieving double-sided bounds. Compared to [5] the algorithm has the same deviation bounds for tori but avoids negative load. Note that for hypercubes, the $\mathcal{O}(\log n)$ deviation bound of [5] is better than ours.

There are not many deviation results for rotor walks for finite graphs. Recall from Section 2.3.3 that [52] shows results for arbitrary graphs but gives weaker bounds than [62] for graph classes such as tori, expanders, and hypercubes. Compared to [62]

[3]One might wonder why we randomize a model which is itself a derandomized version of the random walk model. To answer this question, it should be noted that our randomized scheme still needs much fewer number of random bits compared to the random walk model, while we leverage randomness to overcome the complexity of analyzing a purely deterministic model.

Table 4.1: Comparison of *D_Propp* with existing deterministic algorithms.

	Deviation Bounds		
Algorithm	Regular Graphs	Hypercubes	Tori ($d = \mathcal{O}(1)$)
Arbitrary Rounding [62]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(n^{1/2d})$
Bounded-Error Rounding [41]	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(1)$
Flow-based Rounding Down [5]	$\mathcal{O}(d)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
D_Propp			
Analysis of [52]	$\mathcal{O}(n\|E\|)$	$\mathcal{O}(\log^3 n)$	$\mathcal{O}(n^2)$
Our Analysis	$\mathcal{O}\left(\frac{d \log n}{1-\lambda}\right)$	$\mathcal{O}(\log^{3/2} n)$	$\mathcal{O}(1)$

our bound of $\mathcal{O}(\log^{3/2} n)$ improves their bound of $\mathcal{O}(\log^2 n)$ for hypercubes (see Table 4.1). For finite constant-degree tori, our deviation bound of $\Theta(1)$ extends the constant deviation results for infinite grids [18] and significantly improves the existing bound of $\mathcal{O}(n^{1/2d})$ in [62].

The immediate question is, whether the bounded-error property results in deviation bounds independent of n for finite graphs other than hypercubes and tori. As we observe in Remark 4.3, this does not hold for finite trees where discrepancies of $\Omega(\sqrt{d \log_d n})$ can arise. This follows from the result of [19] on infinite regular trees (see also Section 2.3.3).

R_Propp. In Section 4.2, we analyze a randomized rotor walk model, called *R_Propp*, as a discrete load balancing scheme with randomized rounding. *R_Propp* distributes all but a $1/(d+1)$ fraction of tokens to the neighbours using a rotor that chooses its initial position randomly at the beginning of every round.

We obtain deviation bounds of $\mathcal{O}(d \log \log n/(1-\lambda))$ for regular graphs, $\Theta(\log n)$ for hypercubes, and $\mathcal{O}(\sqrt{\log n})$ for tori. All the bounds hold *with high probability*. *R_Propp* achieves the same discrepancy bounds as the process of [10] where the extra tokens are sent to randomly chosen neighbours *without* replacement. We noticed that the same discrepancy bounds as the ones in [10] hold when the extra tokens are sent to neighbours

Table 4.2: Comparison of *R_Propp* with existing randomized algorithms.

	Random Bits per node	**deviation Bounds**		
Algorithm	**per round**	Regular Graphs	Hypercubes	Tori ($d = \mathcal{O}(1)$)
Rounding Independently [41]	$\mathcal{O}(d)$	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$	$\mathcal{O}(\log^2 n \log\log n)$	$\mathcal{O}(n^{\frac{1}{2d}} \log\log n)$
Flow-based Independent Rounding [5]	$\mathcal{O}(d)$	$\mathcal{O}(\sqrt{d \log n})$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$
Rounding by Sampling [10]	$\mathcal{O}(d \log d)$	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{\log n})$
R_Propp	$\mathcal{O}(\log d)$	$\mathcal{O}\left(\frac{d \log\log n}{1-\lambda}\right)$	$\Theta(\log n)$	$\mathcal{O}(\sqrt{\log n})$

chosen randomly *with* replacement[4]. Hence, our algorithm achieves the same results as both processes (with and without replacement), but with a smaller amount of randomization (see Table 4.2). Compared to [5, 41], *R_Propp* does not create negative load. Our results for hypercubes and tori improve the results of [41]. Note that the deviation bounds of [5] are better for large d or large λ.

Bounded-Error Processes

For a discrete process $\mathcal{P}$ based on a diffusion matrix $\mathbf{P}$, the *directional rounding error* on edge (i, j) in round t is defined as:

$$e_{i\to j}(t) := x_i^{\mathcal{P}}(t) \cdot \mathbf{P}_{i,j} - y_{i,j}^{\mathcal{P}}(t).$$

We define the rounding error on edge (i, j) in round t as

$$e_{i,j}(t) := e_{i\to j}(t) - e_{j\to i}(t).$$

We also define

$$\Lambda_{i,j}(t) := \sum_{\tau=1}^{t} e_{i,j}(\tau),$$

[4]In this case, the difference between the rounded value and the original value can be more than one.

as the *accumulated rounding error* on edge (i, j) in round t, which is the sum of all the rounding errors on (i, j) up to the end of round t.

Definition 4.1 (Bounded-Error Process)**.** *A discrete process is said to have the* bounded-error property *if for some constant α, and all $(i, j) \in E$ and $t \geqslant 0$, the inequality $\left|\Lambda_{i,j}(t)\right| \leqslant \alpha$ holds regardless of the network topology and initial load distribution.*

4.1 The Deterministic Propp Process

In this section, we introduce and analyze the *D_Propp* process. *D_Propp* mimics the continuous diffusion process *RW(2)*. *RW(2)* is similar to a lazy random walk with loop probabilities of 1/2; i.e., it is governed by the matrix $\mathbf{P}$, where $\mathbf{P}_{i,i} = 1/2$ for every node i, and for all pairs of adjacent nodes i, j we have $\mathbf{P}_{i,j} = 1/(2d)$. When not clear from the context, we use the superscript $\mathcal{D}$ for *D_Propp*.

Algorithm *D_Propp*. Our algorithm works in rounds, in every round every node i distributes all its tokens among itself and its neighbours. A round, in turn, consists of steps. In each step, i assigns one token to itself or a neighbour. Hence, if a node i has load $x_i(t)$ at the beginning of round t, from the viewpoint of i the round consists of $x_i(t)$ steps.

The rotor of each node i has $2d$ states $0, \ldots, 2d-1$. The states $0, \ldots, d-1$ point to the neighbours of i in an arbitrary fixed order, and the rest of the states point to i itself. Let $r_i(t)$ be the state of i's rotor in the beginning of round t. Let $r_i(t, s)$ and $\widehat{r}_i(t, s)$ denote the state of i's rotor in the beginning of step s of round t, and its target node, respectively. In each step, a rotor assigns one token to the node it is pointing to, and then it moves the pointer to the subsequent state (Equations (4.2) and (4.3)). Once every token is assigned, the actual load transfer is performed at the end of the round synchronously (Equation (4.5)).

After feeding $2d\lfloor x_i(t)/2d \rfloor$ tokens to the rotor, each state is visited $\lfloor x_i(t)/2d \rfloor$ times and the rotor reaches the same state $r_i(t)$ as it was in the beginning of the round. At this point, $(x_i(t) \bmod 2d)$ tokens are still left. Let us call these tokens *extra tokens*. A node $j \in N(i)$ receives an extra token if and only if one of the $(x_i(t) \bmod 2d)$ rotor states starting from $r_i(t)$ points to j (Equation (4.4)).

Our analysis of the *D_Propp* process in Section 4.1.1 shows the following results:

$$r_i^{\mathcal{D}}(t) = (r_i^{\mathcal{D}}(t-1) + x_i^{\mathcal{D}}(t-1)) \bmod 2d \tag{4.1}$$

$$r_i^{\mathcal{D}}(t,1) = r_i^{\mathcal{D}}(t) \tag{4.2}$$

$$r_i^{\mathcal{D}}(t,s+1) = (r_i^{\mathcal{D}}(t,s) + 1) \bmod 2d \tag{4.3}$$

$$y_{i,j}^{\mathcal{D}}(t) = \begin{cases} \left\lfloor \frac{x_i^{\mathcal{D}}(t)}{2d} \right\rfloor + 1 & \text{if } \widehat{r}_i^{\mathcal{D}}(t,s) = j, \text{ for some } s \text{ between 1 and} \\ & (x_i^{\mathcal{D}}(t) \bmod 2d) \\ \left\lfloor \frac{x_i^{\mathcal{D}}(t)}{2d} \right\rfloor & \text{otherwise.} \end{cases} \tag{4.4}$$

$$x_i^{\mathcal{D}}(t+1) = x_i^{\mathcal{D}}(t) - \sum_{(i,j)\in E} (y_{i,j}^{\mathcal{D}}(t) - y_{j,i}^{\mathcal{D}}(t)) \tag{4.5}$$

Figure 4.1: The *D_Propp* diffusion process: Equations of round t.

Theorem 4.1. *The D_Propp diffusion scheme exhibits the bounded-error property.*

Let the variable T denote the convergence time of *RW(2)*, and r be a constant. Then Theorem 4.1 yields:

Corollary 4.2. *The deviation between D_Propp and RW(2) (hence, the discrepancy at every round $t \geqslant T$) is $\mathcal{O}(\log^{3/2} n)$ for hypercubes and $\mathcal{O}(1)$ for r-dimensional tori.*

According to Theorem 4.1, rotor walks have bounded-error property on *any* graph, while Corollary 4.2 gives deviation results only for torus and hypercube. This raises the question whether the bounded-error property results in deviation bounds independent of n for finite graphs other than hypercubes and tori. For trees, Remark 4.3 answers this question negatively:

Remark 4.3. *As proved in [19][5], for an arbitrary parameter t, there exist initial token distribution and rotor directions such that in round t a high deviation is enforced at the*

[5] *The following theorem (quoted from [19] with slight changes to match the terminology of the thesis) was stated for Propp machines following random walks with zero loop probabilities:*

Theorem 4.4 ([19, Theorem 3]). *For any initial direction of the rotors and any $T > 0$, there is an initial configuration such that the deviation between the Propp machine and the rotor walk model after T time steps is $\Omega(\sqrt{dT})$.*

root (Section 2.3.3). Though the originial construction of [19] is for infinite trees, we note that it can be truncated at depth t and still cause the same deviation at the root. This yields a deviation lower bound of $\Omega(\sqrt{d \log_d n})$ which depends on the order of the graph.

4.1.1 Analysis of *D_Propp*

To prove the bounded-error property for *D_Propp*, we first derive a formula for the edge errors in the observation below (recall that $\{a\} := a - \lfloor a \rfloor$).

Observation 4.5. *For arbitrary $(i,j) \in E$ and $t \geqslant 0$ we have:*

$$e_{i\to j}(t) = \begin{cases} \left\{\frac{x_i^{\mathcal{D}}(t)}{2d}\right\} - 1 & \text{if } \widehat{r}_i(t,s) = j, \text{ for some } s \text{ between } 1 \text{ and } (x_i^{\mathcal{D}}(t) \bmod 2d) \\ \left\{\frac{x_i^{\mathcal{D}}(t)}{2d}\right\} & \text{otherwise.} \end{cases}$$

Proof. Recall $e_{i\to j}(t)$ is defined as $x_i(t) \cdot \mathbf{P}_{i,j} - y_{i,j}(t)$, which yields:

$$e_{i\to j}(t) = \frac{x_i^{\mathcal{D}}(t)}{2d} - y_{i,j}^{\mathcal{D}}(t).$$

The proof now follows from Equation (4.4). □

We define the *directional* accumulated error on edge (i,j) in round t as

$$\Lambda_{i\to j}(t) := \sum_{\tau=1}^{t} e_{i\to j}(\tau).$$

The key ingredient of our analysis is switching from the global round-oriented view of the process to a local view from an arbitrarily fixed node. Define

$$S_i(t) := \sum_{\tau=1}^{t} x_i^{\mathcal{D}}(\tau)$$

with $S_i(0) := 0$. For every edge (i,j), we define $\lambda_{i,j}(t)$ inductively as follows. Define $\lambda_{i,j}(0) := 0$ and for every k, $S_i(t) < k \leqslant S_i(t+1)$ define:

$$\lambda_{i,j}(k) := \begin{cases} \lambda_{i,j}(k-1) + \frac{1}{2d} - 1 & \text{if } \widehat{r}_i(t, k - S_i(t)) = j, \\ \lambda_{i,j}(k-1) + \frac{1}{2d} & \text{otherwise.} \end{cases} \tag{4.6}$$

The intuition behind this definition is that when allocating a single token we expect that every neighbour receives $1/2d$ tokens on average. The amount received in reality is either zero or one token. Hence, at each step the newly added error is either $1/2d - 1$ or $1/2d$. We note that

Observation 4.6. *For arbitrary $(i,j) \in E$ and arbitrary t, we have $|\lambda_{i,j}(t)| < 1$.*

Proof. From the definition of $\lambda_{i,j}(t)$, it follows that for arbitrary $(i,j) \in E$, $\lambda_{i,j}(.)$ is a periodic function with period $2d$ and amplitude $1 - 1/2d$, with $\lambda_{i,j}(0) = 0$. Thus

$$\max_t \lambda_{i,j}(t) - \min_t \lambda_{i,j}(t) = 1 - \frac{1}{2d} < 1,$$

where

$$\max_t \lambda_{i,j}(t) \geqslant 0$$

and

$$\min_t \lambda_{i,j}(t) \leqslant 0.$$

Therefore we must have

$$\min_t \lambda_{i,j}(t) > -1$$

and

$$\max_t \lambda_{i,j}(t) < 1,$$

which yields $|\lambda_{i,j}(t)| < 1$, as required. □

The following lemma provides a method for translating $\Lambda_{i\to j}(.)$ to $\lambda_{i,j}(.)$:

Lemma 4.7. *For arbitrary edge (i,j) and $t \geqslant 0$ we have:*

$$\Lambda_{i\to j}(t) = \lambda_{i,j}(S_i(t)).$$

Proof. The proof is by induction on t. The base case $\Lambda_{i\to j}(0) = \lambda_{i,j}(0) = 0$ is trivial. It suffices to prove that $\lambda_{i,j}(S_i(k)) - \Lambda_{i\to j}(k) = 0$ holds, assuming $\Lambda_{i\to j}(k-1) = \lambda_{i,j}(S_i(k-1))$. The assumption yields:

$$\begin{aligned} \lambda_{i,j}(S_i(k)) - \Lambda_{i\to j}(k) &= \lambda_{i,j}(S_i(k)) - (\Lambda_{i\to j}(k-1) + e_{i\to j}(k)) \\ &= \lambda_{i,j}(S_i(k)) - \lambda_{i,j}(S_i(k-1)) - e_{i\to j}(k). \end{aligned} \tag{4.7}$$

We express $\lambda_{i,j}(S_i(k)) - \lambda_{i,j}(S_i(k-1))$ in Equation (4.7) as a telescoping sum. Observe that $\lambda_{i,j}(.)$ is a periodic sequence with period $2d$. Hence, for arbitrary integers $q, \ell \geqslant 0$ we have:

$$\sum_{s=q+1}^{q+2d\ell} (\lambda_{i,j}(s) - \lambda_{i,j}(s-1)) = \lambda_{i,j}(q+2d\ell) - \lambda_{i,j}(q) = 0.$$

This is used to simplify the summation below by separating a multiple of $2d$ terms (the second sum) from it:

$$\begin{aligned}
&\lambda_{i,j}(S_i(k)) - \lambda_{i,j}(S_i(k-1)) \\
&= \sum_{s=S_i(k-1)+1}^{S_i(k)} (\lambda_{i,j}(s) - \lambda_{i,j}(s-1)) \\
&= \sum_{s=S_i(k-1)+1}^{S_i(k)-2d\lfloor x_i(k)/(2d)\rfloor} (\lambda_{i,j}(s) - \lambda_{i,j}(s-1)) + \sum_{s=S_i(k)-2d\cdot\lfloor x_i(k)/(2d)\rfloor+1}^{S_i(k)} (\lambda_{i,j}(s) - \lambda_{i,j}(s-1)),
\end{aligned} \tag{4.8}$$

Note that the second sum is 0. We now consider two cases:

(I) For some $1 \leqslant s \leqslant (x_i(k) \bmod 2d)$, we have $\widehat{r}_i(k,s) = j$.

(II) Case (I) does not happen.

In case (I), we use Equation (4.6) and Observation 4.5 to write:

$$\begin{aligned}
\lambda_{i,j}(S_i(k)) - \lambda_{i,j}(S_i(k-1)) &= \sum_{s=S_i(k-1)+1}^{S_i(k-1)+(x_i(k) \bmod 2d)} (\lambda_{i,j}(s) - \lambda_{i,j}(s-1)) \\
&= \frac{1}{2d} \cdot ((x_i(k) \bmod 2d) - 1) + \left(\frac{1}{2d} - 1\right) \\
&= \left\{\frac{x_i(k)}{2d}\right\} - 1 \quad = \; e_{i\to j}(k),
\end{aligned} \tag{4.9}$$

Similarly, in case (II) as well, we get

$$\lambda_{i,j}(S_i(k)) - \lambda_{i,j}(S_i(k-1)) = e_{i\to j}(k).$$

Combined with Equations (4.7) and (4.9), this finishes the proof. □

Proof of Theorem 4.1. We show that $|\Lambda_{i,j}(t)| < 2$ for all $t \geqslant 0$ and $(i,j) \in E$. Recall that $\Lambda_{i,j}(t) := \sum_{\tau=1}^{t} e_{i,j}(\tau)$, where $e_{i,j}(\tau) = e_{i\to j}(\tau) - e_{j\to i}(\tau)$. Thus we have

$$\Lambda_{i,j}(t) = \Lambda_{i\to j}(t) - \Lambda_{j\to i}(t).$$

Hence, to prove $|\Lambda_{i,j}(t)| < 2$, it suffices to show $|\Lambda_{i\to j}(t)| < 1$. For the sake of contradiction, suppose for some $t_1 \geqslant 0$ and $(i_1, j_1) \in E$, we have $|\Lambda_{i_1\to j_1}(t_1)| \geqslant 1$. By Lemma 4.7, it holds that

$$\Lambda_{i_1\to j_1}(t_1) = \lambda_{i_1,j_1}(S_{i_1}(t_1));$$

therefore $|\lambda_{i_1,j_1}(S_{i_1}(t_1))| \geqslant 1$. This contradicts Observation 4.6, and the proof follows. □

Proof of Corollary 4.2. We apply Theorems A.5 and A.6 by Friedrich et al. [41] to Theorem 4.1. This shows that the deviation between *D_Propp* and *RW(2)* is $\mathcal{O}(\log^{3/2} n)$ for hypercube, and $\mathcal{O}(1)$ for r-dimensional torus with $r = \mathcal{O}(1)$, as required.

□

4.2 The Randomized Propp Process

In this section, we analyze a randomized rotor walk, called *R_Propp*, which randomly repositions the rotors at the beginning of each round. We view *R_Propp* as a randomized rounding diffusion process. *R_Propp* mimics the standard diffusion process *RW(d + 1)*; *RW(d + 1)* is similar to a random walk with self-loop probability of $1/(d+1)$, governed by the diffusion matrix $\mathbf{P}$, where for all neighbouring nodes $i, j, \mathbf{P}_{i,j} = 1/(d+1)$[6].

We use superscripts $\mathcal{R}$ for *R_Propp*, $\mathcal{S}$ for standard diffusion and capitalized letters for loads and flows, to emphasize these variables are now random. When clear from the context, we omit the superscripts.

Algorithm *R_Propp*. The algorithm works, similar to *D_Propp*, in rounds consisting of one step for each token. Each rotor has $d + 1$ states, corresponding to the nodes in $N(i) \cup \{i\}$. In the beginning of each round, each rotor is directed to a random state (Equation (4.10)). Let $R_i(t)$ be the state of i's rotor in the beginning of round t. Let $R_i(t, s)$ denote the state of i's rotor in the beginning of step s of round t, and and let $\widehat{R}_i(t, s)$ be the node it is pointing to. Upon feeding it a token, a rotor assigns the token to the node it is pointing to, and moves to the subsequent state (Equations (4.11) and (4.12)). A node j receives an extra token if and only if one of the $(x_i(t) \bmod (d+1))$ rotor states starting from $R_i(t)$ points to j (Equation (4.13)). Once every token is assigned, the actual load transfer is performed at the end of each round synchronously (Equation (4.14)).

Note that, similar to *D_Propp*, *R_Propp* avoids the negative load which can occur due to independent rounding in the randomized rounding algorithms of [5, 41, 39], while achieving the same bounds on the discrepancy (double-sided).

[6]Here, there is no technical limitation for choosing this particular loop probability; it can be chosen differently, and the proofs can be adjusted accordingly without changing the asymptotic bounds. In contrast to *D_Propp* in which deviation bounds hold only with loop probability of 1/2, in *R_Propp* we chose this particular loop probability to be more similar to other randomized algorithms.

$$R_i^{\mathcal{R}}(t) = \text{ an integer in } [0,d] \text{ chosen independently and uniformly at random;} \tag{4.10}$$

$$R_i^{\mathcal{R}}(t,1) = R_i^{\mathcal{R}}(t) \tag{4.11}$$

$$R_i^{\mathcal{R}}(t,s+1) = (R_i^{\mathcal{R}}(t,s)+1) \bmod (d+1) \tag{4.12}$$

$$Y_{i,j}^{\mathcal{R}}(t) = \begin{cases} \left\lfloor \frac{X_i^{\mathcal{R}}(t)}{(d+1)} \right\rfloor + 1 & \text{if } \widehat{R}_i^{\mathcal{R}}(t,s) = j, \text{ for some } s \text{ between 1 and} (X_i^{\mathcal{R}}(t) \bmod (d+1)) \\ \left\lfloor \frac{X_i^{\mathcal{R}}(t)}{(d+1)} \right\rfloor & \text{otherwise.} \end{cases} \tag{4.13}$$

$$X_i^{\mathcal{R}}(t+1) = X_i^{\mathcal{R}}(t) - \sum_{(i,j)\in E} (Y_{i,j}^{\mathcal{R}}(t) - Y_{j,i}^{\mathcal{R}}(t)) \tag{4.14}$$

Figure 4.2: The *R_Propp* diffusion process: Equations of round t.

To analyze *R_Propp*, we first consider general graphs (Theorem 4.8), and then we show improved bounds for hypercubes and tori (Theorem 4.9). In both theorems, the variable T denotes the convergence time of *RW(d+1)*. We assume that the discrepancy of the initial distribution is $\mathcal{O}(\exp(n^\kappa))$ for some constant κ.

Theorem 4.8. *The deviation between R_Propp and RW(d+1) (hence, the discrepancy at $t = T$) is $\mathcal{O}(d \log\log n/(1-\lambda))$ w.h.p.*

Note that for d-regular expander, the deviation between *R_Propp* and *RW(d + 1)* is $\mathcal{O}(d \log\log n)$.

Theorem 4.9. *The deviation between R_Propp and RW(d+1) (hence, the discrepancy at time $t = T$) is w.h.p., $\mathcal{O}(\sqrt{\log n})$ if G is an r-dimensional torus with $r = \mathcal{O}(1)$ and $\mathcal{O}(\log n)$ if G is a hypercube. Furthermore, the hypercube bound is tight.*

4.2.1 Definitions and Basic Facts

For arbitrary node k at round t, let

$$E_k(t) := X_k^{\mathcal{R}}(t) - x_k^{\mathcal{S}}(t)$$

denote the difference between the loads of *R_Propp* and *RW(d + 1)*. It is known [62] that

$$E_k(t+1) = \sum_{s=0}^{t} \sum_{(i,j)\in E} E_{i,j}(t-s)\left(\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k}\right).$$

For our proofs, we slightly reformulate this expression:

$$E_k(t+1) = \sum_{s=0}^{t} \sum_{i\in V} \sum_{j\in N(i)} E_{i\to j}(t-s)\left(\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k}\right).$$

For each edge (i, j) we define the Bernoulli random variable $Z^{(t)}_{i,j}$ which is one if an extra token is sent from i to j in round t and zero otherwise. More formally,

$$Z_{i,j}(t) := \begin{cases} 1 & \text{if } \widehat{R}^{\mathcal{R}}_i(t,s) = j, \text{ for some } s \text{ between 1 and } (X^{\mathcal{R}}_i(t) \bmod (d+1)) \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

The rounding error on edge (i, j) in round t, denoted by $E_{i\to j}(t)$ is obtained by the following formula (recall that $\{a\} := a - \lfloor a \rfloor$):

$$E_{i\to j}(t) := \left\{ \frac{X^{\mathcal{R}}_i(t)}{d+1} \right\} - Z^{(t)}_{i,j} \tag{4.16}$$

Thus we have:

$$X^{\mathcal{R}}_i(t+1) = X^{\mathcal{R}}_i(t) + \sum_{j\in N(i)} \left(\frac{X^{\mathcal{R}}_j(t) - X^{\mathcal{R}}_i(t)}{d+1} + E_{i\to j}(t) - E_{j\to i}(t) \right). \tag{4.17}$$

We now observe the following facts:

Observation 4.10. *For every edge (i, j) and round t, we have*

1. $\mathbf{Ex}\left[Z^{(t)}_{i,j} \;\middle|\; X^{\mathcal{R}}(t) \right] = \left\{ \frac{X^{\mathcal{R}}_i(t)}{d+1} \right\}$;
2. $\mathbf{Ex}\left[E_{i\to j}(t) \;\middle|\; X^{\mathcal{R}}(t) \right] = 0$.

Proof. To prove the first part, we simply note that Equation (4.15) implies $Z_{i,j}(t)$ is 1 with probability $(X^{\mathcal{R}}_i(t) \bmod (d+1))/(d+1)$ and 0 otherwise. To prove the second part, we write:

$$\mathbf{Ex}\left[E_{i\to j}(t) \;\middle|\; X^{\mathcal{R}}(t) \right] = \mathbf{Ex}\left[\left\{ \frac{X^{\mathcal{R}}_i(t)}{d+1} \right\} - Z^{(t)}_{i,j} \;\middle|\; X^{\mathcal{R}}(t) \right] \quad \text{(Using Equation (4.16))}$$

$$= \left\{ \frac{X_i^{\mathcal{R}}(t)}{d+1} \right\} - \mathbf{Ex}\left[Z_{i,j}^{(t)} \;\middle|\; X^{\mathcal{R}}(t) \right]$$
$$= 0 \qquad \text{(Using the first part)}$$

□

4.2.2 A General Bound for Arbitrary Graphs

In this section we provide an analysis of *R_Propp* for general graphs, proving Theorem 4.8.

The Proof Outline. To bound $E_k(t+1)$, we first find a probabilistic bound on the error on arbitrary node k and round t. Fix a node k and round t. Define

$$\mathcal{S}_k(s) := \sum_{i \in V} \sum_{j \in N(i)} E_{i \to j}(t-s) \left(\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s \right). \tag{4.18}$$

We break the summation at $t_0 := \min\left\{ \frac{C \log\log n}{1-\lambda}, t \right\}$, where $C \geqslant 1$ is an arbitrary constant:

$$|E_k(t+1)| \leqslant \left| \sum_{s=0}^{t_0-1} \mathcal{S}_k(s) \right| + \left| \sum_{s=t_0}^{t} \mathcal{S}_k(s) \right| \tag{4.19}$$

It is easy to bound the first sum deterministically (Observation 4.11). To bound the second sum, we first use the method of averaged bounded differences (Theorem A.4) to obtain a concentration result for $\mathcal{S}_k(s)$ for arbitrary s. We consider the Doob sequence of the function $\mathcal{S}_k(s)$ with respect to the sequence of randomly chosen rotor states in round s (Lemma 4.12). A similar technique is used in [10], but their function and random variables span several rounds and the Lipschitz constants are estimated quite differently.

In Lemma 4.13 we bound each $\mathcal{S}_k(s)$ by a properly chosen δ_s. Then we argue that the whole sum is bounded by the sum of δ_s values w.h.p.

The first summation of Equation (4.19) can be easily bounded as follows:

Observation 4.11. *For arbitrary node k we have*

$$\left| \sum_{s=0}^{t_0-1} \mathcal{S}_k(s) \right| \leqslant 2\ Cd\ \log\log n/(1-\lambda).$$

Proof. We have

$$\left| \sum_{i \in V} \sum_{j \in N(i)} E_{i \to j}(t-s) \left(\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s \right) \right| = \left| \sum_{s=0}^{t_0-1} \sum_{(i,j) \in E} E_{i,j}(t-s)\ \left(\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s \right) \right|$$

$$
\begin{aligned}
&\leqslant \sum_{s=0}^{t_0-1} \sum_{(i,j)\in E} |E_{i,j}(t-s)| \left|\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k}\right| \\
&\leqslant \sum_{s=0}^{t_0-1} \sum_{(i,j)\in E} \left|\mathbf{P}^s_{i,k}\right| + \left|\mathbf{P}^s_{j,k}\right| \\
&\leqslant 2\,d \sum_{s=0}^{t_0-1} \sum_{i\in V} \left|\mathbf{P}^s_{i,k}\right| \\
&= 2\,d \sum_{s=0}^{t_0-1} 1 \\
&= 2\ C\ d \log\log n/(1-\lambda).
\end{aligned}
$$

□

Now we give the bound for a fixed round:

Lemma 4.12. *For fixed node k, round s, and $\delta > 0$,*

$$\mathbf{Pr}\left[\mathcal{S}_k(s) \geqslant \delta\right] \leqslant 2\exp\left(-\delta^2/16\ d^2\lambda^{2s}\right).$$

Proof. Note that by the definition of $\mathcal{S}_k(s)$ (Equation (4.18)), for a fixed $X^{\mathcal{R}}(t-s)$, the function $\mathcal{S}_k(s)$ depends only on the randomly chosen rotor states $R_i(t-s)$, for $1 \leqslant i \leqslant n$. We represent these random variables as a sequence $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n$, such that for any $\ell \geqslant 1$, we have $\boldsymbol{U}_\ell = R_\ell(t-s)$. Let $\boldsymbol{U}_{\boldsymbol{\ell}}$ denote $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_\ell$. Consider the martingale sequence $\mathbf{Ex}\left[\mathcal{S}_k(s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}}\right], 1 \leqslant \ell \leqslant n$. To apply the method of averaged bounded differences (Theorem A.4), we first provide a bound on the following expression:

$$c_\ell := \left|\mathbf{Ex}\left[\mathcal{S}_k(s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[\mathcal{S}_k(s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right| \tag{4.20}$$

We have:

$$
\begin{aligned}
&\mathbf{Ex}\left[\mathcal{S}_k(s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[\mathcal{S}_k(s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1}\right] \\
&\leqslant \sum_{i\in V} \sum_{j\in N(i)} \left(\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k}\right) \cdot \\
&\qquad \Big(\mathbf{Ex}\left[E_{i\to j}(t-s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[E_{i\to j}(t-s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\Big) \\
&\leqslant \sum_{i\in V} \sum_{j\in N(i)} \left(\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k}\right) \cdot
\end{aligned}
$$

$$\Bigg(\mathbf{Ex}\left[\left\{ \frac{X_i(t-s)}{d+1} \right\} - Z_{i,j}(t-s) \;\middle|\; X(t-s), \boldsymbol{U_\ell} \right] -$$
$$\mathbf{Ex}\left[\left\{ \frac{X_i(t-s)}{d+1} \right\} - Z_{i,j}(t-s) \;\middle|\; X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1} \right] \Bigg)$$
$$\leqslant \sum_{i \in V} \sum_{j \in N(i)} \left(\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k} \right) \cdot$$
$$\Big(\mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U_\ell} \right] - \mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1} \right] \Big) \tag{4.21}$$
$$= \sum_{i \in V} \sum_{j \in N(i)} \left(\mathbf{P}^s_{i,k} - \mathbf{P}^s_{j,k} \right) \cdot \Delta_{i,j}(t-s), \tag{4.22}$$

where

$$\Delta_{i,j}(t-s) := \mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U_\ell} \right] - \mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1} \right].$$

Note that Equation (4.21) holds since conditioned on $X(t-s)$ the variables $X_i(t-s)$ are obtained deterministically.

To bound Equation (4.22) we consider $\sum_{j \in N(i)} \left| \Delta_{i,j}(t-s) \right|$ for $i = \ell$ and $i \neq \ell$ separately:

Case $\boldsymbol{i = \ell}$. Let

$$b := (X_\ell(t-s) \bmod (d+1))$$

be the number of extra tokens of ℓ in round $t-s$. Knowing $\boldsymbol{U}_\ell$ determines the b destinations of the extra tokens, namely $j_1, \ldots, j_b \in N(i) \cup \{i\}$. Prior to this, for all $j \in N(i)$ we had

$$\mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1} \right] = b/(d+1)$$

After $\boldsymbol{U}_\ell$ is determined, we have $\mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U_\ell} \right] = 1$ if $j \in \{j_1, \ldots, j_b\}$, and $\mathbf{Ex}\left[Z_{i,j}(t-s) \mid X(t-s), \boldsymbol{U_\ell} \right] = 0$ otherwise. Thus, $\Delta_{i,j}(t-s) = 1 - b/(d+1)$ if $j \in \{j_1, \ldots, j_b\} - \{i\}$ and $\Delta_{i,j}(t-s) = -b/(d+1)$ otherwise.

Case $\boldsymbol{i \neq \ell}$. In this case $\boldsymbol{U}_\ell$ corresponds to $R_\ell(t-s)$, which is the randomly chosen rotor state on node ℓ in round $t-s$. Given $X(t-s)$, the random variable $Z_{i,j}(t-s)$ is independent of $\boldsymbol{U}_\ell$. Hence,

$$\sum_{j \in N(i)} \left| \Delta_{i,j}(t-s) \right| = \sum_{j \in N(i)} \left| \mathbf{Ex}\left[\, Z_{i,j}(s) \mid \, X(t-s), \boldsymbol{U_\ell} \, \right] - \mathbf{Ex}\left[\, Z_{i,j}(s) \mid \, X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}-1} \, \right] \right|$$
$$= 0.$$

Combining the two cases, we obtain:

$$\begin{aligned}
&\sum_{i\in V}\sum_{j\in N(i)} \Delta_{i,j}(t-s)\cdot\left(\mathbf{P}^s_{i,k}-\mathbf{P}^s_{j,k}\right)\\
&=\sum_{j\in N(\ell)} \Delta_{\ell,j}(t-s)\cdot\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)\\
&=\sum_{j\in\{j_1,\ldots,j_b\}} \Delta_{\ell,j}(t-s)\cdot\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)+\sum_{j\in N(\ell)\setminus\{j_1,\ldots,j_b\}} \Delta_{\ell,j}(t-s)\cdot\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)\\
&=\sum_{j\in\{j_1,\ldots,j_b\}} (1-b/(d+1))\cdot\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)+\sum_{j\in N(\ell)\setminus\{j_1,\ldots,j_b\}} (-b/(d+1))\cdot\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right).
\end{aligned} \tag{4.23}$$

Combining Equations (4.20), (4.22), and (4.23), and using the Cauchy–Schwarz inequality, we have:

$$\begin{aligned}
c_\ell^2 &\leqslant d\cdot\max\left\{(1-b/(d+1))^2,(-b/(d+1))^2\right\}\cdot\sum_{j\in N(\ell)}\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)^2\\
&\leqslant d\cdot\sum_{j\in N(\ell)}\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)^2.
\end{aligned}$$

To apply Theorem A.4, we consider $\sum_{1\leqslant\ell\leqslant n} c_\ell^2$:

$$\begin{aligned}
\sum_{1\leqslant\ell\leqslant n} c_\ell^2 &\leqslant d\cdot\sum_{1\leqslant\ell\leqslant n}\sum_{j\in N(\ell)}\left(\mathbf{P}^s_{\ell,k}-\mathbf{P}^s_{j,k}\right)^2\\
&\leqslant 4\ d\cdot\sum_{1\leqslant\ell\leqslant n}\sum_{j\in N(\ell)}\left(\mathbf{P}^s_{\ell,k}-\frac{1}{n}\right)^2\\
&\leqslant 8\ d^2\lambda^{2s}. \qquad \text{(Lemma A.7)}
\end{aligned} \tag{4.24}$$

Using Equation (4.18), Observation 4.10 and linearity of expectation, we get

$$\mathbf{Ex}\left[\mathcal{S}_k(s)\mid X(t-s)\right]=0.$$

Therefore, by Theorem A.4, for any $\delta\geqslant 0$ we have:

$$\mathbf{Pr}\left[|\mathcal{S}_k(s)|\geqslant\delta\mid X(t-s)\right]\leqslant 2\exp\left(-\delta^2/16\ d^2\lambda^{2s}\right). \tag{4.25}$$

Applying the law of total probability to Equation (4.25), completes the proof. □

Lemma 4.13. *For arbitrary node k we have:*

$$\mathbf{Pr}\left[\left|\sum_{s=t_0}^{t}\mathcal{S}_k(s)\right|\geqslant\frac{8\ d\ \sqrt{7C}}{1-\lambda}\right]=\mathcal{O}(n^{-2C}).$$

Proof. We prove the bound by providing different bounds for different elements of the sum, bounding each $\mathcal{S}_k(s)$ by a properly chosen δ_s. Define $\delta_s := 4\ d\ \sqrt{7C \log n \lambda^s}$. We have:

$$\begin{aligned}
\sum_{s=\frac{\log\log n}{1-\lambda}}^{t} \delta_s &= \sum_{s=\frac{\log\log n}{1-\lambda}}^{t} 4d\ \sqrt{7C\log n}\lambda^{s/2} \\
&\leqslant 4\ d\ \sqrt{7C\log n}\ \frac{\lambda^{\log\log n/(2(1-\lambda))}}{1-\sqrt{\lambda}} \\
&\leqslant 4\ d\ \sqrt{7C\log n}\ \frac{1/\sqrt{\log n}}{1-\sqrt{\lambda}} \\
&= 4\ d\ \sqrt{7C}\ \frac{1+\sqrt{\lambda}}{1-\lambda} \\
&\leqslant \frac{8\ d\ \sqrt{7C}}{1-\lambda} \qquad \text{(Since } \lambda \leqslant 1\text{).} \qquad (4.26)
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}
\mathbf{Pr}\left[\left|\sum_{s=t_0}^{t} \mathcal{S}_k(s)\right| \geqslant \frac{8d\ \sqrt{7C}}{1-\lambda}\right] &\leqslant \sum_{s=\frac{\log\log n}{1-\lambda}}^{t} \mathbf{Pr}\left[|\mathcal{S}_k(s)| \geqslant \delta_s\right] \\
&\leqslant 2\ n^{-6C} \sum_{s=1}^{\infty} \frac{\lambda^s}{\log n} \\
&= \mathcal{O}(n^{-2C}),
\end{aligned}$$

where we use $1/(1-\lambda) = \mathcal{O}(n^4)$ [10, Lemma 2.1]. □

Proof of Theorem 4.8. Now, using Equation (4.19), Observation 4.11 and Lemma 4.13 we get for arbitrary $1 \leqslant k \leqslant n$ and arbitrary constant $C \geqslant 1$, with probability at least $1 - \mathcal{O}(n^{-2C})$,

$$\begin{aligned}
|E_k(t+1)| &\leqslant \frac{Cd\ \log\log n}{1-\lambda} + \frac{8\ d\ \sqrt{7C}}{1-\lambda} \\
&= \mathcal{O}\left(\frac{d\ \log\log n}{1-\lambda}\right). \qquad (4.27)
\end{aligned}$$

Using the results in [62], we have $T = \mathcal{O}(\log(Kn)/1-\lambda) = \mathcal{O}(n^{\kappa+5})$. We choose $C \geqslant \kappa + 6$. By the union bound we have for all k and $t \leqslant T$

$$|E_k(t+1)| = \mathcal{O}(d\ \log\log n/(1-\lambda))$$

with probability at least $1 - \mathcal{O}(n^{-C})$. □

4.2.3 Graph-specific Bounds

In this section we prove Theorem 4.9 which for hypercubes and tori improves the bounds of Theorem 4.8.

The Proof Outline. We provide a probabilistic bound on $E_k(t+1)$ for arbitrarily fixed round t and node k. We follow the martingale argument of [10] (with slight changes) up to deriving the expression for the sum of squared Lipschitz constants, which in our case has an additional factor of d compared to that of [10]. Though this is not an issue when $d = \mathcal{O}(1)$ (e.g. constant degree tori (Lemma 4.14)), for hypercubes we have to reformulate the sum to a new expression (Equation (4.28)) and estimate it in a different way compared to [10], so that we obtain the same deviation bound of $\mathcal{O}(\log n)$ (Lemma 4.15).

Proof of Theorem 4.9 . Fix arbitrary round t and node k. The variable $E_k(t+1)$ depends on the random rotor states in the first t rounds. We denote these variables by $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_{tn}$, such that for any $\ell \geqslant 1$, we have

$$\boldsymbol{U}_\ell := R_i(s) \text{ iff } \ \ell = (s-1)\ n + i.$$

We apply the method of averaged bounded differences (Theorem A.4) on the martingale $\mathbf{Ex}\left[E_k(t+1) \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right], (1 \leqslant \ell \leqslant tn)$, where $\boldsymbol{U}_{\boldsymbol{\ell}}$ is again defined as $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_\ell$. Let

$$c_\ell := \left| \mathbf{Ex}\left[E_k(t+1) \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[E_k(t+1) \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right] \right|.$$

Following steps generally similar to [10], we write:

$$\sum_{\ell=1}^{tn} c_\ell^2 = \sum_{s=0}^{t} \sum_{i=1}^{n} \left(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s \right) \right)^2, \tag{4.28}$$

where we used

$$\Delta_{i,j}(s) := \mathbf{Ex}\left[Z_{i,j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[Z_{i,j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]$$

with $\ell = (s-1)\,n + i$. By Observation 4.10, linearity of expectation, and the law of total probability, we get

$$\mathbf{Ex}\left[E_k(t+1)\right] = 0.$$

By Theorem A.4, if we choose $\delta^2 = (\alpha + \kappa + 6) \log n \cdot \sum_{\ell=1}^{tn} c_\ell^2$ for arbitrary $\alpha > 0$, then for every node k and a fixed round $t \leqslant T$, with probability $1 - \mathcal{O}(n^{-(\alpha+\kappa+6)})$ we have

$|E_k(t+1)| \leqslant \delta$. Since $T = \mathcal{O}(n^{\kappa+5})$, with probability $1 - \mathcal{O}(n^{-\alpha})$, for every $k \in V$ and round $t \leqslant T$, we have

$$|E_k(t+1)| \leqslant \delta.$$

As $\sum_{\ell=1}^{tn} c_\ell^2$ is of $\mathcal{O}(1)$ for torus and $\mathcal{O}(\log n)$ for hypercube (Lemmas 4.15 and 4.14), the proof holds.

We now establish the tightness of the bound for hypercubes. To do this, we show that there is an initial load vector for the hypercube with $d = 2^\kappa - 1$ dimensions ($\kappa \geqslant 1$), such that the deviation between *R_Propp* and *RW(d + 1)* is $\Omega(\log n)$ with probability $1 - n^{\Omega(1)}$.

The construction uses the notion of *2-vertex colouring (2-VC)* of graphs. In a 2-VC, no two nodes with distance of at most two have the same colour. The minimum number of colours required by any 2-VC of G is denoted by $\chi_{\bar{2}}(G)$. It is known [70] that for a d-dimensional hypercube, $d + 1 \leqslant \chi_{\bar{2}}(G) \leqslant 2^{\lceil \log_2(d+1) \rceil}$. Hence, if $d = 2^\kappa - 1$, we have

$$\chi_{\bar{2}}(G) = d + 1.$$

Let $\{C_1, \ldots, C_{d+1}\}$ denote the partitioning of V induced by the colour assignment, and

$$V_1 := C_1 \cup \ldots \cup C_{(d+1)/2}.$$

We choose $X^0()$ such that $X_0^i() = (d+1)/2$ if $i \in V_1$ and $X_i(0) = 0$ otherwise. *RW(d+1)* sends a load of $1/2$ from every node in V_1 to its neighbours in the first round. For arbitrary node i, the set $\{i\} \cup N(i)$ contains exactly $(d+1)/2$ such nodes. Hence, *RW(d + 1)* balances the load in one round. In *R_Propp*, an arbitrary node i receives a token on the edge $(i, j), j \in V_1$ with probability $1/2$. For fixed i these events are independent, thus we have:

$$\begin{aligned}
\mathbf{Pr}\left[X_i(1) \geqslant \frac{3}{8}(d+1)\right] &\geqslant \binom{(d+1)/2}{3(d+1)/8} 2^{-(d+1)/2} \\
&\geqslant \left(\frac{4}{3}\right)^{3(d+1)/8} \cdot 2^{-(d+1)/2} \\
&\geqslant n^{-1+\alpha},
\end{aligned}$$

for some constant $\alpha > 0$. To achieve independence, we consider only a subset of V with pairwise distance of at least 3. Since all nodes have degree $\log n$, there exists $s \subset V$ of size at least $n/\log^3 n$ with this property. This yields:

$$\mathbf{Pr}\left[\exists u \in s : X_u(1) \geqslant \frac{3}{8}(d+1)\right] \geqslant 1 - (1 - n^{-1+\alpha})^{|s|}$$

$$\geqslant 1 - n^{-\beta},$$

for some constant β. Also, in *RW(d+1)* for arbitrary node u we have $x_2^{\square}() = (d+1)/4$. Hence, with probability at least $1 - n^{-\beta}$, *R_Propp* has deviation of $(d+1)/8 = \Omega(\log n)$ from *RW(d + 1)*.

□

Lemma 4.14. *For an r-dimensional torus with $r = \mathcal{O}(1)$, we have $\sum_{\ell=1}^{tn} c_\ell^2 = \mathcal{O}(1)$.*

Proof.

$$\begin{aligned}\sum_{\ell=1}^{tn} c_\ell^2 &= \sum_{s=0}^{t}\sum_{i=1}^{n}\Big(\sum_{j\in N(i)} \Delta_{i,j}(t-s)\cdot\big(\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s\big)\Big)^2 \\ &\leqslant \sum_{s=0}^{t}\sum_{i=1}^{n}\Big(\max_{j\in N(i)}\big|\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s\big|\cdot\sum_{j\in N(i)}\big|\Delta_{i,j}(t-s)\big|\Big)^2. \qquad (4.29)\end{aligned}$$

Consider an arbitrary round s and node i. Let $b = (X_\ell(s) \bmod (d+1))$ be the number of extra tokens of i in round s and $\boldsymbol{U}_\ell$ with $\ell = (s-1)n + i$ be the randomly chosen rotor state of node i in round s. Knowing $\boldsymbol{U}_\ell$ determines the b destinations of the extra tokens, namely $j_1, \ldots, j_b \in N(i) \cup \{i\}$. Prior to knowing $\boldsymbol{U}_\ell$, for all $j \in N(i)$, we had

$$\mathbf{Ex}\left[Z_{i,j}(s) \mid X(s), \boldsymbol{U}_{\boldsymbol{\ell}-1}\right] = b/(d+1).$$

After $\boldsymbol{U}_\ell$ is determined, we have $\mathbf{Ex}\left[Z_{i,j}(s) \mid X(s), \boldsymbol{U}_{\boldsymbol{\ell}}\right] = 1$ if $j \in \{j_1, \ldots, j_b\} - \{i\}$, and $\mathbf{Ex}\left[Z_{i,j}(s) \mid X(t-s), \boldsymbol{U}_{\boldsymbol{\ell}}\right] = 0$ otherwise. Hence, we get

$$\sum_{j\in N(i)}\big|\Delta_{i,j}(t-s)\big| = \mathcal{O}(d).$$

Therefore, it follows from Equation (4.29) that:

$$\begin{aligned}\sum_{\ell=1}^{tn} c_\ell^2 &\leqslant \mathcal{O}(d^2)\sum_{s=0}^{t}\sum_{i=1}^{n}\Big(\max_{j\in N(i)}\big|\mathbf{P}_{i,k}^s - \mathbf{P}_{j,k}^s\big|\Big)^2 \\ &= \mathcal{O}\big((\Upsilon_2(G))^2\big) \\ &= \mathcal{O}(1), \qquad \text{(By [10, Theorem 4.2])}\end{aligned}$$

where

$$\Upsilon_2(G) := \max_{k\in V}\left(\frac{1}{2}\sum_{t=0}^{\infty}\sum_{i=1}^{n}\max_{j\in N(i)}\big|\mathbf{P}_{i,k}^t - \mathbf{P}_{j,k}^t\big|^2\right)^{1/2}$$

is the *refined local divergence* measure introduced in [10]. □

Lemma 4.15. *For a hypercube,* $\sum_{\ell=1}^{tn} c_\ell^2 = \mathcal{O}(d)$.

Proof. Without loss of generality, let $k = 0^d$ (for simplicity, we denote 0^d by 0). It suffices to prove that the following three statements hold for a properly chosen $t' = \mathcal{O}\left(\frac{\log d}{1-\lambda_2}\right) = \mathcal{O}(d \log d)$:

(1) $\sum_{s=t'}^{\infty} \sum_{i=1}^{n} \left(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right) \right)^2 = \mathcal{O}(1)$,

(2) $\sum_{s=0}^{t'-1} \sum_{\|i\| \geqslant 5} \left(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right) \right)^2 = \mathcal{O}(d)$,

(3) $\sum_{s=0}^{t'-1} \sum_{\|i\| \leqslant 4} \left(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right) \right)^2 = \mathcal{O}(d)$,

where $\|i\|$ denotes the distance of node i from node 0^d.

By symmetry of the hypercube, for any two nodes i, j with $\|i\|_1 = \|j\|_1$ and any round s, we have $\mathbf{P}_{i,0}^s = \mathbf{P}_{j,0}^s$. Thus, we can define $\mathbf{P}_h^s$ to denote $\mathbf{P}_{i,0}^s$ for arbitrary node i with $\|i\| = h$.

Proof of statement (1).

$$
\begin{aligned}
&\sum_{s=t'}^{\infty} \sum_{i} \left(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right) \right)^2 \\
&\leqslant d \sum_{s=t'}^{\infty} \sum_{i} \sum_{j \in N(i)} \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right)^2 \quad \text{(Cauchy–Schwarz)} \\
&\leqslant d \sum_{s=t'}^{\infty} d\lambda_2^{2s} \\
&= \mathcal{O}(1), \qquad \text{for a properly chosen } t' = \mathcal{O}\left(\frac{\log d}{1-\lambda_2} \right)
\end{aligned}
$$

Proof of statement (2). By the Cauchy–Schwarz inequality,

$$
\begin{aligned}
&\sum_{s=0}^{t'-1} \sum_{\|i\| \geqslant 5} \left(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right) \right)^2 \\
&\leqslant d \sum_{s=0}^{t'-1} \sum_{\|i\| \geqslant 5} \sum_{j \in N(i)} \left(\mathbf{P}_{i,0}^s - \mathbf{P}_{j,0}^s \right)^2 \\
&\leqslant 2\, d^2 \sum_{h=4}^{d-1} \binom{d}{h+1} \cdot \sum_{s=1}^{t'-1} \left(\mathbf{P}_h^s - \mathbf{P}_{h+1}^s \right)^2
\end{aligned}
$$

$$\leqslant 2\,d^2 \sum_{h=4}^{d-1} \binom{d}{h+1} \left(\sum_{s=1}^{t'-1} \left(\mathbf{P}_h^s - \mathbf{P}_{h+1}^s \right) \right)^2 . \tag{4.30}$$

where the last inequality holds since by Lemma A.10 $\mathbf{P}_h^s \geqslant \mathbf{P}_{h+1}^s$. We split the outer sum into two parts, first considering $4 \leqslant h \leqslant d-4$. We have:

$$\begin{aligned}
&\sum_{h=4}^{d-4} \binom{d}{h+1} \left(\sum_{s=1}^{t'-1} \left(\mathbf{P}_h^s - \mathbf{P}_{h+1}^s \right) \right)^2 \\
&\leqslant \sum_{h=4}^{d-4} \binom{d}{h+1} \left(\sum_{s=0}^{\infty} \left(\mathbf{P}_h^s - \mathbf{P}_{h+1}^s \right) \right)^2 && \text{(Lemma A.10)} \\
&= \sum_{h=4}^{d-4} \frac{\binom{d}{h+1}}{\binom{d}{h}^2} \left(\frac{1}{n} \cdot \frac{d+1}{d-h} \sum_{\ell=h+1}^{d} \binom{d}{\ell} \right)^2 && \text{(Lemmas A.10 and A.9)} \\
&= \sum_{h=4}^{d-4} \frac{d-h}{h+1} \frac{1}{\binom{d}{h}} \left(\frac{1}{n} \cdot \frac{d+1}{d-h} \sum_{\ell=h+1}^{d} \binom{d}{\ell} \right)^2 \\
&\leqslant \sum_{h=4}^{d-4} \frac{d-h}{h+1} \frac{1}{\binom{d}{h}} \left(\frac{d+1}{d-h} \right)^2 && \text{Since } \sum_{\ell=h+1}^{d} \binom{d}{\ell} \leqslant n \\
&= \sum_{h=4}^{d-4} \frac{(d+1)^2}{(h+1)(d-h)} \frac{1}{\binom{d}{h}} \\
&\leqslant d \cdot \sum_{h=4}^{d-4} \frac{1}{\binom{d}{h}} && \text{(For large } n\text{)} \\
&= d \cdot \left(\sum_{h=4}^{d/2} \frac{1}{\binom{d}{h}} + \sum_{h=d/2+1}^{d-4} \frac{1}{\binom{d}{d-h}} \right) \\
&= 2\,d \cdot \sum_{h=4}^{d/2} \frac{1}{\binom{d}{h}} \\
&\leqslant 2\,d \cdot \left(\sum_{h=4}^{\sqrt{d}} \left(\frac{h}{d} \right)^h + \sum_{h=\sqrt{d}}^{d/2} \left(\frac{h}{d} \right)^h \right) \\
&\leqslant 2\,d \cdot \left(\sum_{h=4}^{\sqrt{d}} (d)^{-h/2} + \sum_{h=\sqrt{d}}^{d/2} 2^{-h} \right) \\
&= 2\,d \cdot \mathcal{O}\left(\frac{1}{d^2} + 2^{-\sqrt{d}} \right) \\
&= \mathcal{O}(1/d). && (4.31)
\end{aligned}$$

For the case $d-3 \leqslant h \leqslant d-1$, we have:

$$\sum_{h=d-3}^{d-1} \frac{d-h}{h+1} \frac{1}{\binom{d}{h}} \left(\frac{1}{n} \cdot \frac{d+1}{d-h} \sum_{\ell=h+1}^{d} \binom{d}{\ell} \right)^2 = \mathcal{O}(d^9/n^2)$$
$$= \mathcal{O}(n^{-1}). \tag{4.32}$$

Combining Equations (4.30), (4.31), and (4.32) the proof of the second statement follows.

Proof of statement (3). We have:

$$\begin{aligned}
&\sum_{s=0}^{t'-1} \sum_{\|i\| \leqslant 4} \Big(\sum_{j \in N(i)} \Delta_{i,j}(t-s) \cdot \left(\mathbf{P}^s_{i,0} - \mathbf{P}^s_{j,0} \right) \Big)^2 \\
&= \sum_{s=0}^{t'-1} \sum_{\|i\| \leqslant 4} \Bigg(\left(\mathbf{P}^s_{\|i\|} - \mathbf{P}^s_{\|i\|+1} \right) \sum_{\substack{j \in N(i), \\ \|j\| = \|i\|+1}} \Delta_{i,j}(t-s) \\
&\quad + \left(\mathbf{P}^s_{\|i\|} - \mathbf{P}^s_{\|i\|-1} \right) \sum_{\substack{j \in N(i), \\ \|j\| = \|i\|-1}} \Delta_{i,j}(t-s) \Bigg)^2 \\
&\leqslant 2 \sum_{s=0}^{t'-1} \sum_{\|i\| \leqslant 4} \left(\mathbf{P}^s_{\|i\|} - \mathbf{P}^s_{\|i\|+1} \right)^2 \Big(\sum_{\substack{j \in N(i), \\ \|j\| = \|i\|+1}} \Delta_{i,j}(t-s) \Big)^2 \\
&+ 2 \sum_{s=0}^{t'-1} \sum_{1 \leqslant \|i\| \leqslant 4} \left(\mathbf{P}^s_{\|i\|} - \mathbf{P}^s_{\|i\|-1} \right)^2 \Big(\sum_{\substack{j \in N(i), \\ \|j\| = \|i\|-1}} \Delta_{i,j}(t-s) \Big)^2
\end{aligned} \tag{4.33}$$

Consider an arbitrary round s and node i. Let

$$b := (X_\ell(s) \bmod (d+1))$$

be the number of extra tokens of i in round s and $\boldsymbol{U}_\ell$ with $\ell = (s-1)\,n + i$ be the randomly chosen rotor state of node i in round s. Observe that $\Delta_{i,j}(s) = 1 - b/(d+1)$ if j receives an extra token and $\Delta_{i,j}(s) = -b/(d+1)$ otherwise. On the other hand, since $\|i\| \leqslant 4$, the set $\{\Delta_{i,j}(s) : j \in N(i), \|j\| = \|i\| - 1\}$ has size at most 4. With each $\Delta_{i,j}(s)$ lying in $(-1, 1)$, we have:

$$\Big| \sum_{\substack{j \in N(i), \\ \|j\| = \|i\|-1}} \Delta_{i,j}(s) \Big| \leqslant \sum_{\substack{j \in N(i), \\ \|j\| = \|i\|-1}} \big| \Delta_{i,j}(s) \big| \leqslant 4.$$

We also observe that $\sum_{j\in N(i)\cup\{i\}} \Delta_{i,j}(s) = 0$, hence:

$$\begin{aligned}\Big|\sum_{\substack{j\in N(i),\\ \|j\|=\|i\|+1}} \Delta_{i,j}(s)\Big| &= \Big|\sum_{\substack{j\in N(i),\\ \|j\|=\|i\|-1}} \Delta_{i,j}(s) + \Delta_{i,i}(s)\Big| \\ &\leqslant \Big|\sum_{\substack{j\in N(i),\\ \|j\|=\|i\|-1}} \Delta_{i,j}(s)\Big| + \big|\Delta_{i,i}(s)\big| \\ &\leqslant 5.\end{aligned}$$

Therefore it follows from Equation (4.33) that:

$$\begin{aligned}&\sum_{s=0}^{t'-1}\sum_{\|i\|\leqslant 4}\left(\sum_{j\in N(i)}\Delta_{i,j}(t-s)\cdot\left(\mathbf{P}^s_{i,0}-\mathbf{P}^s_{j,0}\right)\right)^2 \\ &\leqslant 5\sum_{s=0}^{t'-1}\sum_{\|i\|\leqslant 4}\left(\mathbf{P}^s_{\|i\|}-\mathbf{P}^s_{\|i\|+1}\right)^2 \\ &\quad+4\sum_{s=0}^{t'-1}\sum_{1\leqslant\|i\|\leqslant 4}\left(\mathbf{P}^s_{\|i\|}-\mathbf{P}^s_{\|i\|-1}\right)^2 \\ &\leqslant 5\sum_{s=0}^{t'-1}\sum_{\|i\|\leqslant 4}\left(\mathbf{P}^s_{\|i\|}\right)^2+4\sum_{s=0}^{t'-1}\sum_{1\leqslant\|i\|\leqslant 4}\left(\mathbf{P}^s_{\|i\|-1}\right)^2 && \text{(Lemma A.10)} \\ &\leqslant 5\sum_{h=0}^{4}\left(\binom{d}{h}+\binom{d}{h+1}\right)\cdot\sum_{s=0}^{t'-1}(\mathbf{P}^s_h)^2 && \text{(Lemma A.10)} \\ &= 5\sum_{h=0}^{4}\binom{d+1}{h+1}\cdot\sum_{s=0}^{t'-1}(\mathbf{P}^s_h)^2\,. && (4.34)\end{aligned}$$

Observe that,

$$\begin{aligned}&\sum_{h=0}^{4}\binom{d+1}{h+1}\cdot\sum_{s=0}^{t'-1}(\mathbf{P}^s_h)^2 \\ &= d+1+\binom{d+1}{1}\cdot\sum_{s=1}^{t'-1}(\mathbf{P}^s_0)^2+\binom{d+1}{2}\cdot\sum_{s=1}^{t'-1}(\mathbf{P}^s_1)^2 \\ &\quad+\sum_{h=2}^{4}\binom{d+1}{h+1}\cdot\sum_{s=0}^{t'-1}(\mathbf{P}^s_h)^2 \\ &\leqslant d+1+(d+1)\cdot\sum_{s=1}^{t'-1}\frac{1}{(d+1)^2}+\binom{d+1}{2}\cdot\left(\sum_{s=1}^{6}(\mathbf{P}^s_1)^2+\sum_{s=7}^{t'-1}(\mathbf{P}^s_1)^2\right)\end{aligned}$$

$$
\begin{aligned}
&+\sum_{h=2}^{4}\binom{d+1}{h+1}\cdot\sum_{s=0}^{t'-1}(\mathbf{P}_h^s)^2\\
&\leqslant d+1+t'/d+\ (d+1)^2\cdot\left(\sum_{s=1}^{6}(\mathbf{P}_1^s)^2+\left(\sum_{s=7}^{t'-1}\mathbf{P}_1^s\right)^2\right)\\
&+\sum_{h=2}^{4}\binom{d+1}{h+1}\cdot\sum_{s=0}^{t'-1}\frac{1}{\binom{d}{h}^2}\qquad\text{(Lemma A.11)}\\
&\leqslant d+1+t'/d+\ (d+1)^2\cdot\left(\sum_{s=1}^{6}\frac{1}{(d+1)^2}+\left(\sum_{s=7}^{t'-1}\mathbf{P}_1^s\right)^2\right)+2\ t'/d\\
&=\mathcal{O}\Big(d+d^2\cdot\Big(\sum_{s=7}^{t'-1}\mathbf{P}_1^s\Big)^2\Big).
\end{aligned}
\tag{4.35}
$$

Now, it suffices to prove $\sum_{s=7}^{t'-1}\mathbf{P}_1^s=\mathcal{O}(d^{-1})$. Observe that in the step t of a random walk starting from i, the probability that the distance from 0^d does not increase is at most $(t+\|i\|)/(d+1)$. Consider a random walk of length 7 starting from an arbitrary node i with $\|i\|=1$. To arrive at a node j with $\|j\|\leqslant 2$ at step 7, the random walk can at most 4 times increase its distance to $0^{\log n}$, hence there must be at least 3 steps in which the distance does not increase: This implies for all j with $\|j\|\leqslant 2$,

$$
\sum_{j:\|j\|\leqslant 2}\mathbf{P}_{i,j}^7\leqslant\binom{7}{3}\cdot\left(\frac{8}{d+1}\right)^3=\mathcal{O}(d^{-3}).
\tag{4.36}
$$

We have:

$$
\begin{aligned}
\sum_{s=7}^{t'-1}\mathbf{P}_1^s&=\sum_{s=7}^{t'-1}\mathbf{P}_{i,0}^s\\
&\leqslant\sum_{s=7}^{t'-1}\left(\sum_{\substack{j\\\|j\|\leqslant 2}}\mathbf{P}_{i,j}^7\cdot\mathbf{P}_{j,0}^{s-7}+\sum_{\substack{j\\3\leqslant\|j\|\leqslant d}}\mathbf{P}_{i,j}^7\cdot\mathbf{P}_{j,0}^{s-7}\right)\\
&\leqslant\sum_{s=7}^{t'-1}\left(\sum_{\substack{j\\\|j\|\leqslant 2}}\mathbf{P}_{i,j}^7\cdot\mathbf{P}_{j,0}^{s-7}+\left(\frac{1}{\binom{d}{3}}\cdot\sum_{\substack{j\\3\leqslant\|j\|\leqslant d/2}}\mathbf{P}_{i,j}^7\right)\right.\\
&\quad\left.+\left(\frac{1}{\binom{d}{d/2}}\cdot\sum_{\substack{j\\d/2\leqslant\|j\|\leqslant d}}\mathbf{P}_{i,j}^7\right)\right)
\end{aligned}
$$

$$\leqslant t' \cdot \left(\mathcal{O}(d^{-3}) \cdot \left(\sum_{\substack{j \\ \|j\| \leqslant 2}} \mathbf{P}_{0,j}^{s-7} \right) + \frac{1}{\binom{d}{3}} \cdot 1 + \frac{1}{\binom{d}{d/2}} \right)$$

$$= \mathcal{O}(d^{-1}). \qquad \text{(Using Equation (4.36))} \tag{4.37}$$

where we have used Lemmas A.11 and A.10 to simplify the expressions. Applying Equation (4.37) to Equation (4.35) and subsequently Equation (4.34) finishes the proof. □

Chapter 5

Discrete Second-Order Processes

In this chapter we show three main results.

(1) We present a general framework for rounding continuous diffusion schemes to discrete schemes. Our approach (Section 5.1) estimates first the error between a continuous diffusion scheme and the rounded discrete version, similar to [62]. Then we combine that error term with martingales techniques (similar to the ones used in [10]) to bound the deviation between the continuous scheme and a discrete scheme based on randomized rounding. Note that the result in [62] are only valid for a class of homogeneous FOS schemes and [10] analyzes a fixed FOS diffusion scheme with specific transition matrix. In contrast, we introduce an error estimation that allows us to show results for a larger class of diffusion algorithms (see Definition 5.2) in heterogeneous networks, including SOS.

(2) We show that randomized SOS has a deviation of $\mathcal{O}\Big(\frac{d\cdot\log s_{\max}\cdot\sqrt{\log n}}{(1-\lambda)^{3/4}}\Big)$, where λ is the second largest eigenvalue and $s_{\max}$ is the maximum speed. See the right half of Table 5.1 for detailed bounds for different networks. The results are worse than the best results for FOS schemes, which is due to a larger upper bound on the refined local divergence (for the definition see Section 5.1.2). Note that the runtime of SOS is much lower than the runtime of FOS, i.e., $\mathcal{O}(\log(Kn)/\sqrt{1-\lambda})$ (assuming optimal β) compared to $\mathcal{O}\left(\log(Kn)/(1-\lambda)\right)$ many rounds for FOS.

(3) We show that the continuous SOS scheme with optimal β will not generate negative load if (at $t=0$) the minimum load of every node is at least $\mathcal{O}\left(\sqrt{n}\cdot\Delta(0)/\sqrt{1-\lambda}\right)$. Here $\Delta(0)$ is the difference between the minimum and maximum load at $t=0$.

For discrete SOS schemes we show a bound of $\mathcal{O}\left(\sqrt{n}\cdot\Delta(0)+d^2/\sqrt{1-\lambda}\right)$. To the best of our knowledge these are the first results specifying the sufficient minimum load to avoid negative load.

Table 5.1: Comparison of deviation bounds of discrete FOS and SOS from their continuous counterparts.

	Arbitrary Rounding		**Randomized Rounding**	
Graph	FOS [62]	SOS (Theorem 5.10)	FOS [10]	SOS (Theorem 5.12)
Ring	$\mathcal{O}(n)$	$\mathcal{O}(n\sqrt{n})$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(n^{3/4}\sqrt{\log n})$
2-D Torus	$\mathcal{O}(\sqrt{n})$	$\mathcal{O}(n)$	$\mathcal{O}(\sqrt{\log n})$	$\mathcal{O}(n^{3/8}\sqrt{\log n})$
Hypercube	$\mathcal{O}(\log^2 n)$	$\mathcal{O}(\sqrt{n}\log^2 n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log^{9/4} n)$
Expander	$\mathcal{O}(\log n)$	$\mathcal{O}(\sqrt{n})$	$\mathcal{O}(\log\log n)$	$\mathcal{O}(\sqrt{\log n})$

5.1 General Framework for FOS Schemes

In this section we first generalize the framework of Rabani et al. [62] to a wider class of processes (see Section 5.1.1) and obtain an equation estimating the deviation of the discrete process from its continuous version. The estimation is valid as long as the continuous process is *linear* (Definition 5.2). In [62] the deviation is expressed in terms of the diffusion matrix. Here, we present an analysis from a different perspective which allows us to obtain essentially the same deviation formula, but for a larger class of processes. Our analysis can be applied to the second order processes and heterogeneous models. In Section 5.1.2 we present the framework that transforms a continuous load balancing process C into a discrete process $R(C)$ using randomized rounding.

For simplicity we consider in this section only first order processes. In the next section we generalize the framework to SOS.

5.1.1 Deviation between Continuous and Discrete FOS Schemes

For $j \in N(i)$, we define $y_{i,j}^{\mathcal{A}}(t)$ as the amount of load sent from i to j in round t (this value is negative if the actual direction is from j to i), where $N(i)$ represents the set of

neighbours of i. $y^A(t)$ is the matrix with $y^{\mathcal{A}}_{i,j}(t)$ as its entry in row i and column j. Recall from Section 1.2 that we use the following definition and notation for *rounding*: each balancing process A is regarded as a function that, given the current state of the network, determines for every edge e and round t the amount of load that has to be transferred over e in t. Hence, we can regard $y^A(t)$ as the result of applying a function A, i.e. $y^A(t) = A(x^{\mathcal{A}}(t))$. Using this we formally define discrete processes:

Definition 5.1. *Let C be a continuous process. A process D is said to be a discrete version of C with rounding scheme R_D if for every vector $\mathbf{x}$, we have $D(\mathbf{x}) = R_D(C(\mathbf{x}))$ where R_D is a function that rounds each entry of the matrix to an integer.*

Although it may not be a necessary condition, our analysis in this section requires the process to exhibit a *linearity* property in the following sense:

Definition 5.2 (linearity)**.** *A diffusion process A is said to be* linear *if for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$ and $a, b \in \mathbb{R}$ we have $A(a\mathbf{x} + b\mathbf{x}') = a \cdot A(\mathbf{x}) + b \cdot A(\mathbf{x}')$.*

Let C be a continuous process and D its discrete version. Let $\widehat{Y}(t)$ represent $C(x^{\mathcal{D}}(t))$. Then we can say that D always *attempts to* set $y^{\mathcal{D}}_{i,j}(t)$ to $\widehat{Y}_{i,j}(t)$. Hence, we call $\widehat{Y}(t)$ the *continuous scheduled load*. We define the *rounding error* as

$$e_{i\to j}(t) := \widehat{Y}_{i,j}(t) - y^{\mathcal{D}}_{i,j}(t).$$

Note that

$$e_{i\to j}(t) = -e_{j\to i}(t).$$

Lemma 5.1. *Both FOS and SOS as defined in Section 2.1 are linear.*

Proof. Let M be the diffusion matrix and $0 \leqslant \beta \leqslant 2$. Observe that both FOS and SOS can be described by the following general equation (see Equations (2.1) and (2.4)):

$$y_{i,j}(t) = (\beta - 1) \cdot y_{i,j}(t-1) + \beta \cdot M_{i,j} \cdot x_i(t) \qquad \text{for } t \geqslant 1, \tag{5.1}$$

Thus the algorithm A – where based on the choice of parameter β, A can represent either FOS and SOS – is defined by

$$A(\mathbf{x}, \mathbf{y}) = (\beta - 1)\mathbf{y} + \beta M\mathbf{x}$$

Let $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n, \mathbf{y}, \mathbf{y}' \in \mathbb{R}^{n\times n}$ and $a, b \in \mathbb{R}$. Then we have

$$A(a\mathbf{x} + b\mathbf{x}', a\mathbf{y} + b\mathbf{y}') = (\beta - 1)\,(a\mathbf{y} + b\mathbf{y}') + \beta M(a\mathbf{x} + b\mathbf{x}')$$

$$= a((\beta-1)\mathbf{y} + \beta M\mathbf{x}) + b((\beta-1)\mathbf{y}' + \beta M\mathbf{x}')$$
$$= aA(\mathbf{x},\mathbf{y}) + bA(\mathbf{x}',\mathbf{y}')$$

which shows that A is linear. □

In the next definition $\widehat{\mathbf{i}}$ denotes the unit vector with 1 as its i'th entry for every $i \leqslant n$.

Definition 5.3 (Contributions)**.** *Let* $\mathbf{x}$ *and* $\mathbf{x}$*' be the load vectors obtained from applying* C *for* t *rounds on* $\widehat{\mathbf{i}}$ *and* $\widehat{\mathbf{j}}$*, respectively. For two fixed nodes* i *and* k *and* $j \in N(i)$ *the* contribution *of edge* (i,j) *on node* k *after* t *rounds is defined as:* $\mathscr{C}^{C}_{k,i\to j}(t) := \mathbf{x}_k - \mathbf{x}'_k$.

The next theorem provides a general form of the FOS deviation formula of [62] which has served as a basis for analyzing several discrete FOS processes.

Lemma 5.2. *Consider a linear diffusion process* C *and its discrete version* D *with an arbitrary rounding scheme. Then, for an arbitrary node* k *and round* t *we have:*

$$x^{\mathcal{D}}_k(t) - x^{\mathcal{C}}_k(t) = \sum_{s=1}^{t} \sum_{\{i,j\}\in E} e_{i\to j}(t-s)\,\mathscr{C}^{C}_{k,i\to j}(s)$$

Proof. Fix a node k and round t. Suppose we *sequentialize* the load balancing actions of the process by imposing an arbitrary ordering on the edges. Then, t rounds in the parallel view is equivalent to $|E|\cdot t$ steps in the sequentialized view. In the following, let $\tau := |E|\cdot t$. With a slight abuse of notation we let $C^\infty \circ D^\ell$ denote a *hybrid* process in which the load balancing actions are determined by D in steps 1 to ℓ, and by C afterwards, where $0 \leqslant \ell \leqslant \tau$. Observe that $x^{\mathcal{C}^\infty\circ\mathcal{D}^\tau}_k(t) = x^{\mathcal{D}}_k(t)$ and $x^{\mathcal{C}^\infty\circ\mathcal{D}'}_k(t) = x^{\mathcal{C}}_k(t)$. Thus we can write $x^{\mathcal{D}}_k(t) - x^{\mathcal{C}}_k(t)$ in the form of a telescoping sum as follows:

$$x^{\mathcal{D}}_k(t) - x^{\mathcal{C}}_k(t) = x^{\mathcal{C}^\infty\circ\mathcal{D}^\tau}_k(t) - x^{\mathcal{C}^\infty\circ\mathcal{D}'}_k(t) = \sum_{\ell=1}^{\tau}\left(x^{\mathcal{C}^\infty\circ\mathcal{D}^\ell}_k(t) - x^{\mathcal{C}^\infty\circ\mathcal{D}^{\ell-\infty}}_k(t)\right) \qquad (5.2)$$

Fix an arbitrary step ℓ and let $\{i,j\}$ and s be the edge and the round corresponding to the step ℓ. Both $C^\infty \circ D^{\ell-1}$ and $C^\infty \circ D^\ell$ start their round $s+1$ with load vectors that are the same except maybe in i and j. This happens because $C^\infty \circ D^{\ell-1}$ forwards $\widehat{Y}_{i,j}(s)$ over $\{i,j\}$ while in $C^\infty \circ D^\ell$ this amount is $\widehat{Y}_{i,j}(s) - e_{i\to j}(s)$. Thus, by the definition of $\mathscr{C}^{C}_{k,i\to j}(t)$ and using the linearity property of the process we get:

$$x^{\mathcal{C}^\infty\circ\mathcal{D}^\ell}_k(t) - x^{\mathcal{C}^\infty\circ\mathcal{D}^{\ell-\infty}}_k(t) = e_{i\to j}(s)\,\mathscr{C}^{C}_{k,i\to j}(t-s),$$

Plugging the above into Equation (5.2) and translating the summation index we get

$$x_k^{\mathcal{D}}(t) - x_k^{\mathcal{C}}(t) = \sum_{s=0}^{t-1} \sum_{\{i,j\}\in E} e_{i\to j}(s)\, \mathscr{C}^C_{k,i\to j}(t-s) = \sum_{s=1}^{t} \sum_{\{i,j\}\in E} e_{i\to j}(t-s)\, \mathscr{C}^C_{k,i\to j}(s)$$

□

5.1.2 Framework for Randomized FOS Schemes

In this section we use Lemma 5.2 to analyze a randomized rounding scheme for a general class of continuous load balancing algorithms. Our technique is based on the results of [10] where the authors analyzed a fixed discrete FOS process for homogeneous d-regular graphs using randomized rounding. Their algorithm is based on a continuous process in which every node sends a $1/(d+1)$-fraction of its load to each neighbour. As before, to denote random variables we capitalize the previously defined notation. Initially, the discrete algorithm rounds $X_i/(d+1)$ down if it is not an integer. This leaves $(d+1)\cdot\lfloor X_i/(d+1)\rfloor$ surplus tokens on node i, which they call *excess* tokens. The excess tokens are then distributed by sending the tokens to neighbours which are uniformly sampled without replacement.

Here we apply the technique in a much more general way, using Lemma 5.2 to express the deviation between the randomized and deterministic algorithm. We introduce a randomized framework that converts a general class of continuous processes to their discrete versions using randomized rounding.

The Randomized Rounding Algorithm.

Fix a node i. For each edge $e=\{i,j\}$ let $\widehat{Y}_{i,j}(t) = C(X^R(t))$ be the load that is sent over e by the continuous process C. The rounding scheme works as follows. First, it rounds $\widehat{Y}_{i,j}(t)$ down for all the edges. This leaves $r := \sum_{j:\widehat{Y}_{i,j}(t)\geqslant 0}\{\widehat{Y}_{i,j}(t)\}$ excess load on node i. Then it takes $\lceil r\rceil$ additional tokens and sends each of them out with a probability of $r/\lceil r\rceil$. With the remaining probability the excess tokens remain on node i. The tokens which do not remain on i are sent to a neighbour j with a probability of $\{\widehat{Y}_{i,j}(t)\}/r$. Let $Z_{i,j}(t)$ be a counting random variable denoting the number of excess tokens that i sends to j in round t. Then we have:

$$Y^{\mathcal{R}}_{i,j}(t) = \begin{cases} \lfloor \widehat{Y}_{i,j}(t)\rfloor + Z_{i,j}(t) & \text{if } \widehat{Y}_{i,j}(t)\geqslant 0 \\ -Y^{\mathcal{R}}_{j,i}(t) & \text{otherwise} \end{cases} \tag{5.3}$$

The deviation bound is expressed based on the *refined local divergence* ($\Upsilon^C(G)$) defined below, which is a function of both the algorithm and the graph:

$$\Upsilon^C(G) := \max_{k\in V}\left(\sum_{s=0}^{\infty}\sum_{i=1}^{n}\max_{j\in N(i)}\left(\mathscr{C}^C_{k,i\to j}(s)\right)^2\right)^{1/2}$$

$\Upsilon^C(G)$ is a generalization of the refined local divergence $\Upsilon(G)$ introduced in [10]. In the next section we show the following result.

Theorem 5.3. *Let C be a continuous FOS and let $R = R(C)$ be a discrete FOS using our randomized rounding transformation. In an arbitrary round t we have w.h.p.:*

$$\left|X^R_k(t) - x^{\mathcal{C}}_k(t)\right| = \mathcal{O}\left(\Upsilon^C(G)\cdot\sqrt{d\log n}\right)$$

The proof of Theorem 5.3 relies on the fact that FOS is a linear process (Lemma 5.1) and hence the estimation of Lemma 5.2 can be used as a basis for the randomized analysis. The proof is similar to the proof of [10], the difference is that we use $\mathscr{C}^C_{k,i\to j}(t)$'s instead of diffusion matrix.

Proof of Theorem 5.3

Proof. We begin the proof of Theorem 5.3 with a simple observation.

Observation 5.4. *The following statements are true: (Recall that $\{a\}$ denotes $a-\lfloor a\rfloor$):*

(1) If $\widehat{Y}_{i,j}(t)\geqslant 0$ then $E_{i\to j}(t) = \{\widehat{Y}_{i,j}(t)\} - Z_{i,j}(t)$;

(2) $\mathbf{Ex}\left[E_{i\to j}(t)\right] = 0$.

The first statement of the theorem holds since by definition, $E_{i\to j}(t) := \widehat{Y}_{i,j}(t) - Y^{\mathcal{R}}_{i,j}(t)$ while $Y^{\mathcal{R}}_{i,j}(t) = \left\lfloor \widehat{Y}_{i,j}(t)\right\rfloor + Z_{i,j}(t)$. For the second statement, first suppose that $\widehat{Y}_{i,j}(t)\geqslant 0$. note that $Z_{i,j}(t)$ can be expressed as a sum of $\lceil r\rceil$ identically distributed Bernoulli random variables each of which is one with probability $(r/\lceil r\rceil)\cdot\left(\{\widehat{Y}_{i,j}(t)\}/r\right)$. Thus we have $\mathbf{Ex}\left[Z_{i,j}(t)\right] = \{\widehat{Y}_{i,j}(t)\}$ and from there (2) follows from (1). In the case $\widehat{Y}_{i,j}(t) < 0$, we have $\widehat{Y}_{j,i}(t) = -\widehat{Y}_{i,j}(t) > 0$. Thus by the first case we have $\mathbf{Ex}\left[E_{j\to i}(t)\right] = 0$ and therefore $\mathbf{Ex}\left[E_{i\to j}(t)\right] = -\mathbf{Ex}\left[E_{j\to i}(t)\right] = 0$. □

Let $F_k := X^R_k(t) - x^{\mathcal{C}}_k(t)$ denote the difference in the load of k in round t of R and C. In the following, we first observe that F_k is zero in expectation, and then show that it is well concentrated around its average.

Observation 5.5. $\mathbf{Ex}\left[F_k\right] = 0$

Proof. The statement follows from Lemma 5.2 and Observation 5.4.(2) by the linearity of expectation. □

As in [10], we are going to use the method of averaged bounded differences to obtain concentration results for the random variable F_k. For a fixed initial load vector $X(0)$ the function F_k depends only on the randomly chosen destinations of the excess tokens. There are t rounds, n nodes, and at most d excess tokens per node per round. Similar to [10] we describe these random choices by a sequence of tnd random variables, $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_{tnd}$. For any ℓ with $1 \leqslant \ell \leqslant tnd$, let (s, i, b) be such that $\ell = s \cdot n \cdot d + (i-1)\, d + b$ (note that (s, i, b) is the ℓ-th largest element in the sequence). Then $\boldsymbol{U}_\ell$ refers to the destination of the b-th excess token of vertex i in round s (if there is one). More precisely,

$$\boldsymbol{U}_\ell := \begin{cases} j & \text{if the } b\text{-th excess token of the vertex } i \text{ in round } s \text{ is sent to } j, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{U}_{\boldsymbol{\ell}}$ denote $\boldsymbol{U}_\ell, \ldots, \boldsymbol{U}_1$. To apply the method of averaged bounded differences, we need to bound the difference sequence below:

$$\left|\mathbf{Ex}\left[F_k \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[F_k \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right|. \tag{5.4}$$

As in [10], we consider a fixed ℓ that corresponds to (s_1, i_1, b_1) in the lexicographic ordering.

To bound Equation (5.4), we write

$$\begin{aligned} c_\ell &:= \left|\mathbf{Ex}\left[F_k \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[F_k \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right| \\ &\leqslant \sum_{s=0}^{t} \sum_{\{i,j\}\in E} \left|\mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right| \cdot \left|\mathscr{C}^C_{k,i\to j}(t-s)\right| \end{aligned}$$

As in [10] we split the sum over s into the three parts $1 \leqslant s < s_1$, $s = s_1$, and $s_1 < s \leqslant t$. In the following we show that the sums over $s < s_1$ and $s > s_1$ are both zero while the part $s = s_1$ is upper bounded by $2 \cdot \max_{j\in N(i_1)} \left|\mathscr{C}^C_{k,i\to j}(t-s)\right|$.

$\underline{\mathbf{s < s_1:}}$ For every $\{i, j\} \in E$, $E_{i\to j}(s)$ is already determined by $\boldsymbol{U}_{\boldsymbol{\ell}-1}$. Hence,

$$\sum_{s=1}^{s_1-1} \sum_{\{i,j\}\in E} \left|\mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right| \cdot \left|\mathscr{C}^C_{k,i\to j}(t-s)\right| = 0. \tag{5.5}$$

$\underline{\mathbf{s = s_1:}}$ In this case, $\boldsymbol{U}_{\ell-1}$ determines $\widehat{Y}_{i,j}(t)$ and $E_{i\to j}(s)$ is only affected by $Z_{i,j}(s)$'s (see Observation 5.4):

$$\begin{aligned}
&\sum_{\{i,j\}\in E} \left| \mathbf{Ex}\left[E_{i\to j}(s)\,\middle|\,\boldsymbol{U}_{\ell}\right] - \mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\ell-1}\right]\right| \cdot \left|\mathscr{C}^{C}_{k,i\to j}(t-s)\right| \\
&= \sum_{i=1}^{n} \sum_{j:\widehat{Y}_{i,j}(t)\geqslant 0} \left(\left| \mathbf{Ex}\left[\{\widehat{Y}_{i,j}(t)\} - Z_{i,j}(t) \mid \boldsymbol{U}_{\ell}\right] - \mathbf{Ex}\left[\{\widehat{Y}_{i,j}(t)\} - Z_{i,j}(t) \mid \boldsymbol{U}_{\ell-1}\right] \right| \right) \\
&\qquad\qquad \cdot \left|\mathscr{C}^{C}_{k,i\to j}(t-s)\right| \\
&= \sum_{\{i,j\}\in E} \left| \mathbf{Ex}\left[Z^{(s)}_{i,j} \mid \boldsymbol{U}_{\ell}\right] - \mathbf{Ex}\left[Z^{(s)}_{i,j} \mid \boldsymbol{U}_{\ell-1}\right] \right| \cdot \left|\mathscr{C}^{C}_{k,i\to j}(t-s)\right| \\
&= \sum_{\{i,j\}\in E} \left|\Lambda^{(s)}_{i,j}\right| \cdot \left|\mathscr{C}^{C}_{k,i\to j}(t-s)\right| \\
&\leqslant \sum_{i=1}^{n} \left(\max_{j\in N(i)} \left|\mathscr{C}^{C}_{k,i\to j}(t-s)\right| \right) \sum_{j\in N(i)} \left|\Lambda^{(s)}_{i,j}\right|, \qquad (5.6)
\end{aligned}$$

where we used $\Lambda^{(s)}_{i,j} := \mathbf{Ex}\left[\, Z^{(s)}_{i,j} \mid \boldsymbol{U}_{\ell}\,\right] - \mathbf{Ex}\left[\, Z^{(s)}_{i,j} \mid \boldsymbol{U}_{\ell-1}\,\right]$ to simplify the notation.

As in [10], to bound Equation (5.6) we consider $\sum_{\{i,j\}\in E} \left|\Lambda^{(s)}_{i,j}\right|$ for $i = i_1$ and $i \neq i_1$ separately.

Case 1: Let $i = i_1$. For each $j \in N(i_1)$, define indicator Bernoulli random variables $I_{u,j}, 1 \leqslant u \leqslant d$, where $I_{u,j}$ is one if the u'th excess token of i_1 in round s_1 goes to j and zero otherwise. Note that $Z^{(s_1)}_{i_1,j} = \sum_{1\leqslant u\leqslant d} I_{u,j}$. Let $r = \sum_{j\in N(i_1)} \left\{\widehat{Y}_{i_1,j}(s_1)\right\}$ so that $\lceil r\rceil \geqslant b_1$ be the number of excess tokens of i_1 in round s_1. Clearly, r and the destinations of the excess tokens considered in the previous rounds, are already determined by $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_{\ell-1}$. The remaining receivers $\boldsymbol{U}_{\ell+1}, \ldots, \boldsymbol{U}_{\ell+r-b_1}$ are chosen independently from $N(i_1)\cup\{i_1\}$. Hence, the choice of $\boldsymbol{U}_\ell$ does not affect the distribution of $I_{u,j}$ except for $u = b_1$, and we have:

$$\begin{aligned}
&\mathbf{Ex}\left[\, Z_{i_1,j}(s_1) \mid \boldsymbol{U}_{\ell}\,\right] - \mathbf{Ex}\left[\, Z_{i_1,j}(s_1) \mid \boldsymbol{U}_{\ell-1}\,\right] \\
&= \mathbf{Ex}\left[\, I_{1,j} + \cdots + I_{\lceil r\rceil,j} \mid \boldsymbol{U}_{\ell}\,\right] - \mathbf{Ex}\left[\, I_{1,j} + \cdots + I_{\lceil r\rceil,j} \mid \boldsymbol{U}_{\ell-1}\,\right] \\
&= \mathbf{Ex}\left[\, I_{b_1,j} \mid \boldsymbol{U}_{\ell}\,\right] - \mathbf{Ex}\left[\, I_{b_1,j} \mid \boldsymbol{U}_{\ell-1}\,\right]
\end{aligned}$$

Let $w \in N(i_1)\cup\{i_1\}$ be the destination of the b_1-th excess token of i_1 in round s_1, that is, $\boldsymbol{U}_\ell = w$ and hence, $\Lambda^{(s_1)}_{i_1,w} = 1 - \{\widehat{Y}_{i_1,w}(t)\}/r$. For any $j \in N(i_1)\setminus\{w\}$ we have $\Lambda^{(s_1)}_{i_1,w} = -\{\widehat{Y}_{i_1,j}(t)\}/r$.

Hence,

$$\begin{aligned}\sum_{j\in N(i_1)}\left|\Lambda^{(s_1)}_{i_1,j}\right| &\leqslant 1-\{\widehat{Y}_{i_1,w}(t)\}/\lceil r\rceil+\sum_{j\in N(i_1)\setminus\{w\}}\{\widehat{Y}_{i_1,j}(t)\}/\lceil r\rceil\\ &\leqslant 1+\sum_{j\in N(i_1)}\{\widehat{Y}_{i_1,j}(t)\}/\lceil r\rceil\\ &\leqslant 2, \end{aligned}\tag{5.7}$$

where the last inequality holds since $\sum_{j\in N(i_1)}\{\widehat{Y}_{i_1,j}(t)\}=r\leqslant\lceil r\rceil$.

Case 2: $i\neq i_1$.

As ℓ corresponds to (s_1,i_1,b_1), the random variable $Z_{i,j}(s_1)$ is independent of $\boldsymbol{U}_\ell$ when conditioned on $\boldsymbol{U}_{\boldsymbol{\ell}-1}$. Hence, similar to [10], we have

$$\sum_{\{i,j\}\in E}\left|\Lambda^{(s_1)}_{i,j}\right|=\sum_{j:\{i,j\}\in E}\left|\mathbf{Ex}\left[\,Z^{(s)}_{i,j}\mid\boldsymbol{U}_{\boldsymbol{\ell}}\,\right]-\mathbf{Ex}\left[\,Z^{(s)}_{i,j}\mid\boldsymbol{U}_{\boldsymbol{\ell}-1}\,\right]\right|=0.$$

Combining Case 1 and Case 2 we obtain

$$\begin{aligned}(5.6)&=\left(\max_{j\in N(i_1)}\left|\mathscr{C}^C_{k,i\to j}(t-s)\right|\right)\sum_{j:\{i_1,j\}\in E}\left|\Lambda^{(s)}_{i_1,j}\right|\\ &\quad+\sum_{i\in V,i\neq i_1}\left(\max_{j\in N(i)}\left|\mathscr{C}^C_{k,i\to j}(t-s)\right|\right)\sum_{\{i,j\}\in E}\left|\Lambda^{(s)}_{i,j}\right|\\ &\leqslant\max_{j\in N(i_1)}\left|\mathscr{C}^C_{k,i\to j}(t-s)\right|\cdot 2+0.\end{aligned}\tag{5.8}$$

$\underline{\mathbf{s}>\mathbf{s_1}:}$ Let $\widetilde{\ell}$ be the largest integer that corresponds to round $s-1$. Since $s>s_1$, we have $s-1\geqslant s_1$ and therefore $\widetilde{\ell}\geqslant\ell$. By the choice of $\widetilde{\ell}$, $\boldsymbol{U}_{\widetilde{\ell}},\ldots,Y_1$ determine the load vector at the end of round s_1, $X^R(s_1)$. By Observation 5.4, we obtain $\mathbf{Ex}\left[E_{i\to j}(s)\mid\boldsymbol{U}_{\widetilde{\ell}},\ldots,\boldsymbol{U}_1\right]=0$, and by the law of total expectation,

$$\begin{aligned}\mathbf{Ex}\left[E_{i\to j}(s)\mid\boldsymbol{U}_{\boldsymbol{\ell}}\right]&=\mathbf{Ex}\left[\mathbf{Ex}\left[E_{i\to j}(s)\mid\boldsymbol{U}_{\widetilde{\ell}},\ldots,Y_1\right]\mid\boldsymbol{U}_\ell,\boldsymbol{U}_{\ell-1},\ldots,Y_1\right]\\ &=\mathbf{Ex}\left[0\mid\boldsymbol{U}_\ell,\boldsymbol{U}_{\ell-1},\ldots,\boldsymbol{U}_1\right]=0.\end{aligned}$$

With the same arguments, $\mathbf{Ex}\left[E_{i\to j}(s)\mid\boldsymbol{U}_{\boldsymbol{\ell}-1}\right]=0$, and therefore

$$\sum_{s=s_1+1}^{t}\sum_{\{i,j\}\in E}\left|\mathbf{Ex}\left[E_{i\to j}(s)\mid\boldsymbol{U}_{\boldsymbol{\ell}}\right]-\mathbf{Ex}\left[E_{i\to j}(s)\mid\boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right|\cdot\left|\mathscr{C}^C_{k,i\to j}(t-s)\right|=0.\tag{5.9}$$

This finishes the case distinction. Combining Equations (5.5), (5.8), and (5.9) for the three cases $s < s_1$, $s = s_1$, and $s > s_1$, similar to [10] we obtain that for every fixed $1 \leqslant \ell \leqslant t \cdot n \cdot d$,

$$\begin{aligned} c_\ell =& \left|\mathbf{Ex}\left[F_k \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[F_k \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right| \\ \leqslant& \sum_{s=0}^{t} \sum_{\{i,j\}\in E} \left|\mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}}\right] - \mathbf{Ex}\left[E_{i\to j}(s) \mid \boldsymbol{U}_{\boldsymbol{\ell}-1}\right]\right| \cdot \left|\mathscr{C}^{C}_{k,i_1\to j}(t-s_1)\right| \\ =& 0 + \max_{j\in N(i_1)} \left|\mathscr{C}^{C}_{k,i_1\to j}(t-s_1)\right| \cdot 2 + 0 \\ =& 2 \cdot \max_{j\in N(i_1)} \left|\mathscr{C}^{C}_{k,i_1\to j}(t-s_1)\right|. \end{aligned}$$

Now we consider $\sum_{\ell=1}^{(t+1)nd} (c_\ell)^2$:

$$\begin{aligned} \sum_{\ell=1}^{(t+1)nd} (c_\ell)^2 &\leqslant \sum_{s=0}^{t}\sum_{i=1}^{n}\sum_{b=1}^{d} \Big(2 \max_{j\in N(i)} \left|\mathscr{C}^{C}_{k,i\to j}(t-s)\right|\Big)^2 \\ &= 4d \sum_{s=0}^{t}\sum_{i=1}^{n} \max_{j\in N(i)} \left(\mathscr{C}^{C}_{k,i\to j}(s)\right)^2 \\ &\leqslant 4d \max_{k\in V} \left(\sum_{s=0}^{\infty}\sum_{i=1}^{n} \max_{j\in N(i)} \left(\mathscr{C}^{C}_{k,i\to j}(s)\right)^2\right) \\ &= 8d \left(\Upsilon^{\mathrm{C}}(G)\right)^2. \end{aligned} \tag{5.10}$$

So by Theorem A.4 we have for any $\delta \geqslant 0$, $\mathbf{Pr}\left[|F_k| > \delta\right] \leqslant 2 \exp\left(-\delta^2 / \left(2\sum_{\ell=1}^{tnd}(c_\ell)^2\right)\right)$. Hence by choosing $\delta := \Upsilon^{C}(G)\sqrt{32d \ln n}$, the probability above gets smaller than $2\,n^{-2}$. Applying the union bound we obtain

$$\mathbf{Pr}\left[\exists k \in V \colon |F_k| > \delta\right] \leqslant 2n^{-1}.$$

This implies

$$\mathbf{Pr}\left[\max_{i,j\in[n]} \left|X_i^R(t) - x_i^{\mathcal{C}}(t)\right| \leqslant \delta\right] \geqslant 1 - 2n^{-1},$$

which finishes the proof. □

Using Theorem 5.3 we can also obtain concrete results for randomized FOS processes as stated in the following theorem:

Theorem 5.6. *Let C be a continuous FOS process and let $R = R(C)$ be a discrete FOS process based on the rounding algorithm applied on C. Then*

(1) $\Upsilon^{\mathrm{C}}(G) = \mathcal{O}\left(\sqrt{\frac{d\cdot\log s_{\max}}{1-\lambda}}\right)$.

(2) For any round t *we have w.h.p.* $\left|X_k^R(t) - x_k^{\mathrm{C}}(t)\right| = \mathcal{O}\left(d\cdot\sqrt{\frac{\log n\cdot\log s_{\max}}{1-\lambda}}\right)$.

Proof. The proof is similar to the proof of Theorem 5.12. We have

$$\begin{aligned}\left(\Upsilon^{\mathrm{FOS}}(G)\right)^2 &= \sum_{t=0}^{\infty}\sum_{i=1}^{n}\max_{j\in N(i)}\left(M_{k,i}^t - M_{k,j}^t\right)^2\\ &\leqslant \sum_{t=0}^{\infty}\sum_{i=1}^{n}\sum_{j\in N(i)}\left(M_{k,i}^t - M_{k,j}^t\right)^2\\ &= \sum_{t=0}^{t_1-1}\sum_{i=1}^{n}\sum_{j\in N(i)}\left(M_{k,i}^t - M_{k,j}^t\right)^2 + \sum_{t=t_1}^{\infty}\sum_{i=1}^{n}\sum_{j\in N(i)}\left(M_{k,i}^t - M_{k,j}^t\right)^2 \qquad (5.11)\end{aligned}$$

On the other hand,

$$\begin{aligned}\sum_{t=0}^{t_1-1}\sum_{i=1}^{n}\sum_{j\in N(i)}\left(M_{k,i}^t - M_{k,j}^t\right)^2 &\leqslant \sum_{t=0}^{t_1}\sum_{i=1}^{n}\sum_{j\in N(i)} 2\left((M_{k,i}^t)^2 + (M_{k,j}^t)^2\right)\\ &\leqslant 4\cdot d\cdot\sum_{t=0}^{t_1-1}\sum_{i=1}^{n}(M_{k,i}^t)^2\\ &\leqslant 4\cdot d\cdot\sum_{t=0}^{t_1-1}\|M^t\,\widehat{\mathbf{k}}\|_2^2\\ &= 4\cdot d\cdot\sum_{t=0}^{t_1-1}\left(\|\widehat{\mathbf{k}}\|_2\cdot\max_i\lambda_i^t\right)^2\\ &= 4\cdot d\cdot\sum_{t=0}^{t_1-1}1\\ &\leqslant 4\cdot d\cdot t_1 \qquad (5.12)\end{aligned}$$

Let $t_1 = (\log s_{\max})/(2-2\lambda)$. Note that $\lambda^{1/(1-\lambda)} \leqslant 1/e$. Then we have

$$\begin{aligned}\sum_{t=t_1}^{\infty}\sum_{i=1}^{n}\sum_{j\in N(i)}\left(M_{k,i}^t - M_{k,j}^t\right)^2 &\leqslant \sum_{t=t_1}^{\infty}\sum_{i=1}^{n}\sum_{j\in N(i)} 2\left(\left(M_{k,i}^t - \frac{s_k}{s}\right)^2 + \left(M_{k,j}^t - \frac{s_k}{s}\right)^2\right)\\ &= 4\cdot d\cdot\sum_{t=t_1}^{\infty}\sum_{i=1}^{n}\left(M_{k,i}^t - \frac{s_k}{s}\right)^2\\ &\leqslant 8\cdot d\cdot s_{\max}\cdot\sum_{t=t_1}^{\infty}\lambda^{2t} \qquad (5.13)\end{aligned}$$

$$\leqslant 8 \cdot d \cdot s_{\max} \cdot \lambda^{2t_1} \cdot \frac{1}{1-\lambda}$$
$$\leqslant \frac{8\,d}{1-\lambda} \tag{5.14}$$

where Equation (5.13) follows from Lemma A.8. Combining Equations (5.11), (5.14), and (5.12) we get

$$\left(\Upsilon^{\mathrm{FOS}}(G)\right)^2 = \mathcal{O}\left(\frac{d \cdot \log s_{\max}}{1-\lambda}\right), \tag{5.15}$$

which proves the first statement. The bound in the second statement follows immediately from statement (1) and Theorem 5.3. □

It should be noted that for the special case of homogeneous networks where for all $j \in N(i)$ we have $M_{i,j} = 1/(\gamma d)$ for $\gamma > 1$, there is a deviation bound independent of λ which is better than the one implied by the general bound of Theorem 5.6. This follows from an improved refined local divergence bound given in [64, Theorem 5.3] for the restricted case mentioned above.

Observation 5.7. *For $M_{i,j} = 1/(\gamma d)$ in homogeneous networks we have*

(1) [64, Theorem 5.3]

$$\Upsilon^{\mathrm{FOS}}(G) \leqslant \sqrt{\frac{\gamma d}{2 - 2/\gamma}}$$

(2) The bound in (1) is minimized for $\gamma = 2$, which yields

$$\left|X_k^R(t) - x_k^{\mathrm{C}}(t)\right| = \mathcal{O}(d\sqrt{\log n}).$$

However, there is no simple adjustment of their approach to the general case of arbitrary diffusion matrices and heterogeneous networks.

5.2 Second-Order Diffusion Processes

In this section we show that after some slight adjustments the framework of Section 5.1 can be applied to second-order processes on heterogeneous networks. All we have to do is to state Definitions 5.2 and 5.3 in a more general way that captures the dependence of SOS on the load transfer of the previous round. It is easy to see that Lemma 5.2 and Theorem 5.3 still hold assuming the new definitions. (Note that by Lemma 5.1, SOS

is linear). If C is a second order process, then $y^C(t)$ is determined based on $x^{\mathcal{C}}(t)$ and $y^C(t-1)$. More formally, $y^C(t) = C(x^{\mathcal{C}}(t), y^C(t-1))$. Thus, the new definitions also incorporate $y^C(t-1)$. Again, we use $\widehat{\mathbf{i}}$ to denote the unit vector with 1 as its i'th entry.

Definition 5.4 (linearity)**.** *A process A is said to be* linear *if for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n, \mathbf{y}, \mathbf{y}' \in \mathbb{R}^{n\times n}$ and $a, b \in \mathbb{R}$ we have $A(a\mathbf{x} + b\mathbf{x}', a\mathbf{y} + b\mathbf{y}') = aA(\mathbf{x}, \mathbf{y}) + bA(\mathbf{x}', \mathbf{y}')$.*

Definition 5.5 (Contributions)**.** *Let $\mathbf{x}(0) = \mathbf{x}'(0) = \widehat{\mathbf{i}}$, $\mathbf{y}(0) = \mathbf{0}_{n\times n}$ and $\mathbf{y}'(0)$ is also all zero except $\mathbf{y}'_{i,j}(0) = 1$, so that $\mathbf{x}(1) = \widehat{\mathbf{i}}, \mathbf{x}'(1) = \widehat{\mathbf{j}}$. Let $\mathbf{x}(t+1)$ and $\mathbf{x}$'(t+1) be the load vectors obtained from applying C for t rounds on $(\mathbf{x}(1), \mathbf{y}(0))$ and $(\mathbf{x}'(1), \mathbf{y}'(0))$, respectively. Then the* contribution *of the edge (i, j) on a node k after t rounds is defined as: $\mathscr{C}^C_{k,i\to j}(t) := \mathbf{x}_k(t) - \mathbf{x}'_k(t)$.*

The contributions are expressed based on a sequence of matrices $Q(t)$ defined below, whose role in error propagation is similar to that of the diffusion matrix in FOS:

$$Q(t) = \begin{cases} \mathbf{I} & \text{if } t = 0 \\ \beta \cdot M & \text{if } t = 1 \\ \beta \cdot M\, Q(t-1) + (1-\beta) \cdot Q(t-2) & \text{if } t \geqslant 2 \end{cases} \tag{5.16}$$

Lemma 5.8. *For $t > 0$, we have $\mathscr{C}^{\mathrm{SOS}}_{k,i\to j}(t) = Q_{k,i}(t-1) - Q_{k,j}(t-1)$.*

Proof. Let $\mathbf{x}(0) = \mathbf{x}'(0) = \widehat{\mathbf{i}}$, $\mathbf{y}(0) = \mathbf{0}_{n\times n}$ and $\mathbf{y}'(0)$ is also all zero except $\mathbf{y}'_{i,j}(0) = 1$, so that $\mathbf{x}(1) = \widehat{\mathbf{i}}, \mathbf{x}'(1) = \widehat{\mathbf{j}}$. Let $\mathbf{x}(t+1)$ and $\mathbf{x}$'(t+1) be the load vectors obtained from applying SOS for t rounds on $(\mathbf{x}(1), \mathbf{y}(0))$ and $(\mathbf{x}'(1), \mathbf{y}'(0))$, respectively. Let $\mathscr{C}^{\mathrm{SOS}}_{i\to j}(t)$ be a vector that has $\mathscr{C}^{\mathrm{SOS}}_{k,i\to j}(t)$ as its k'th entry, for $1 \leqslant k \leqslant n$. Let $\mathbf{e} := \widehat{\mathbf{i}} - \widehat{\mathbf{j}}$. Then we have:

$$\mathscr{C}^{\mathrm{SOS}}_{i\to j}(t) = \mathbf{x}(t) - \mathbf{x}'(t) = \begin{cases} \mathbf{0} & \text{if } t = 0 \\ \mathbf{e} & \text{if } t = 1 \\ \beta \cdot M\, \mathscr{C}^{\mathrm{SOS}}_{i\to j}(t-1) + (1-\beta) \cdot \mathscr{C}^{\mathrm{SOS}}_{i\to j}(t-2) & \text{if } t \geqslant 2 \end{cases} \tag{5.17}$$

where the third equation holds because for all $t \geqslant 2$, both $\mathbf{x}(t)$ and $\mathbf{x}'(t)$ follow the same equation $x(t) = \beta \cdot Mx(t-1) + (1-\beta) \cdot x(t-2)$. Now, it can be proved by induction that $\mathscr{C}^{\mathrm{SOS}}_{i\to j}(t) = Q(t-1)\,\mathbf{e}$. Recall that all entries of $\mathbf{e}$ are zero except $\mathbf{e}_i = 1$ and $\mathbf{e}_j = -1$. Therefore, for $t > 0$ we get $\mathscr{C}^{\mathrm{SOS}}_{k,i\to j}(t) = Q_{k,i}(t-1) - Q_{k,j}(t-1)$. □

The following lemma provides a bound for the second norm of $Q(t)$, which is later used in the proofs of Theorems 5.10 and 5.12.

Lemma 5.9. *Let $\beta = \beta_{opt} = 2/(1+\sqrt{1-\lambda^2})$. The following statements are true:*

(1) Eigenvectors of $Q(t)$ form a basis for $\mathbb{R}^n$.

(2) Let $\gamma := \left(\sqrt{\beta-1}\right)^t (t+1)$. Then γ is an upperbound on the eigenvalues of $Q(t)$ except the eigenvalue corresponding to the eigenvector $(s_1, \ldots, s_n)$.

(3) $Q(t)$ has equal column sums.

(4) Define $q(t) := \sum_{1 \leqslant j \leqslant n} Q_{i,j}(t)$ for an arbitrary $1 \leqslant i \leqslant n$ (note that by the statement (3), this is a valid definition). Fix a $1 \leqslant k \leqslant n$, and let the vector $\mathbf{a}$ be such that $\mathbf{a}_i := Q_{k,i}(t) - \frac{s_k}{s} \cdot q(t)$. Then we have: $\|\mathbf{a}\|_2^2 \leqslant 2\, s_{\max}(\beta-1)^t(t+1)^2$.

Proof. Proof of (1). First we observe that the eigenvectors of $Q(t)$ are the same as the eigenvectors of M. This can be proved by an induction using the recurrence of Equation (5.17). Also, note that $M = I - LS^{-1}$ where L is the Laplacian matrix of the graph and S is the diagonal matrix of speeds. The eigenvectors of M are the same as those of LS^{-1}. By [29, proof of Lemma 1] the eigenvectors of LS^{-1} form a basis for $\mathbb{R}^n$. Therefore the eigenvectors of M and the eigenvectors of $Q(t)$ form a basis for $\mathbb{R}^n$.

Proof of (2). From the induction in the proof of statement (1) one can see that corresponding to each eigenvalue λ_j of M an eigenvalue $\gamma_j(t)$ of $Q(t)$ can be obtained according to the following recursion:

$$\gamma_j(t) = \begin{cases} 1 & \text{if } t = 0 \\ \beta\lambda_j & \text{if } t = 1 \\ \beta\lambda_j \cdot \gamma_j(t-1) + (1-\beta)\cdot\gamma_j(t-2) & \text{if } t \geqslant 2 \end{cases} \tag{5.18}$$

Solving the above recursion we get

$$\gamma_j(t) = \begin{cases} \frac{1-(\beta-1)^{t+1}}{2-\beta} & \text{if } \lambda_j = 1, \\ \left(\sqrt{\beta-1}\right)^t (t+1) & \text{if } |\lambda_j| = \lambda, \\ r^t\left(\cos(\theta t) + \sin(\theta t)\cdot\frac{\lambda_j}{\sqrt{\lambda^2-\lambda_j^2}}\right) & \text{if } |\lambda_j| < \lambda, \end{cases} \tag{5.19}$$

where $r = \sqrt{\beta-1}$, and $0 < \theta < \pi$ is such that $\sin\theta = \sqrt{\lambda^2-\lambda_j^2}/\lambda$, and $\cos\theta = \lambda_j/\lambda$. Note that the eigenvalue correspnding to $\lambda_j = 1$ belongs to the eigenvector $(s_1, \ldots, s_n)$.

Hence, it suffices to prove that in Equation (5.19) the case $|\lambda_j| < \lambda$ does not produce eigenvalues bigger than those obtained in the case $|\lambda_j| = \lambda$. Note that

$$\begin{aligned}\gamma_j(t) &= r^t \left(\cos(\theta t) + \sin(\theta t) \cdot \frac{\lambda_j}{\sqrt{\lambda^2 - \lambda_j^2}} \right) \\ &\leqslant \left(\sqrt{\beta - 1}\right)^t \cdot \frac{\sin((t+1)\theta)}{\sin\theta} \\ &\leqslant \left(\sqrt{\beta - 1}\right)^t \cdot (t+1),\end{aligned} \tag{5.20}$$

where in Equation (5.20) we use $\sin(nx) \leqslant n \sin x$ for $0 < x < \pi$ and $n \in \mathbb{N}$.

Proof of (3). The statement follows from a simple induction using Equation (5.17).

Proof of (4). Let $\widehat{\mathbf{a}} := \widehat{\mathbf{k}} - \frac{s_k}{s} \cdot \mathbf{1}_n$. Note that $\mathbf{a} = \widehat{\mathbf{a}}\, Q(t)$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the eigenvectors of $Q(t)$ with eigenvalues $\gamma_1, \ldots, \gamma_n$, and γ be defined as in the statement (2) of the lemma. Using the fact that $M = I - LS^{-1}$, it can be proved by induction that $S^{-1}Q(t)$ is symmetric. Hence, for each right eigenvector $\mathbf{v}_i$ of $Q(t)$ there is a left eigenvector $\mathbf{u}_i = S^{-1}\mathbf{v}_i$ with the same eigenvalue γ_i as proved in the following:

$$(Q(t))^T \mathbf{u}_i = (Q(t))^T S^{-1} S \mathbf{u}_i = (S^{-1}Q(t))^T \mathbf{v}_i = S^{-1}Q(t)\mathbf{v}_i = \gamma_i S^{-1}\mathbf{v}_i = \gamma_i \mathbf{u}_i$$

Also, note that $S^{-1}Q(t)S\mathbf{u}_i = S^{-1}Q(t)\mathbf{v}_i = \gamma_i S^{-1}\mathbf{v}_i = \gamma_i \mathbf{u}_i$. As a result, $\mathbf{u}_i$'s are eigenvectors of $S^{-1}Q(t)S$, which is symmetric because it is the product of symmetric matrices $S^{-1}Q(t)$ and S. Therefore, $\mathbf{u}_1, \ldots, \mathbf{u}_n$ form an orthonormal basis; so we can write $\widehat{\mathbf{a}} = \sum_{i=1}^n c_i \mathbf{u}_i$. Now we write $\mathbf{a} = \widehat{\mathbf{a}}\, Q(t) = \sum_{i=1}^n c_i \mathbf{u}_i Q(t) = \sum_{i=1}^n \gamma_i c_i \mathbf{u}_i$. Therefore,

$$\|\mathbf{a}\|_2^2 = \sum_{i=1}^n \gamma_i^2 c_i^2 \|\mathbf{u}_i\|_2^2 \leqslant \gamma^2 \sum_{i=1}^n c_i^2 \|\mathbf{u}_i\|_2^2 = \gamma^2 \|\widehat{\mathbf{a}}\|_2^2 \tag{5.21}$$

where the inequality uses the fact that $\widehat{\mathbf{a}} \perp (s_1, \ldots, s_n)$ and part (2) of the lemma, and the last equality follows from the fact that $\mathbf{u}_i$'s form an orthonormal basis. Also,

$$\|\widehat{\mathbf{a}}\|_2^2 \leqslant n \cdot \frac{s_k^2}{s^2} + 1 \leqslant \frac{n\, s_k^2}{(n - 1 + s_k)^2} + 1 \leqslant \frac{n\, s_k^2}{2(n-1)s_k} + 1 \leqslant s_k + 1 \leqslant 2 s_{\max}$$

Together with Equation (5.21), this yields $\|\mathbf{a}\|_2^2 \leqslant 2\, s_{\max} (\beta - 1)^t (t+1)^2$. □

5.2.1 Deviation between Continuous and Discrete SOS Schemes

In this section we show a bound on the deviation between a continuous SOS and its rounded version. The authors of [30] show a similar bound on the deviation using the second norm, i.e., they show a bound of $||x^{\mathcal{D}(\mathrm{SOS})}(t)-x^{\mathrm{SOS}}(t)||_2 = \mathcal{O}\left(d\sqrt{ns_{\max}}/(1-\lambda)\right)$.

Theorem 5.10. *Consider a discrete SOS process $D = D(\mathrm{SOS})$ with optimal β and a rounding scheme that rounds a fractional value to either its floor or its ceiling. Then for arbitrary $t \geqslant 0$ we have: $\left|x_k^{\mathcal{D}}(t) - x_k^{\mathrm{SOS}}(t)\right| = \mathcal{O}\left(d\sqrt{ns_{\max}}/1-\lambda\right)$.*

Note that the deviation bound of $\mathcal{O}(d\sqrt{s_{\max}\log n/(1-\lambda)})$ for the FOS is smaller.

Proof. We use Lemma 5.2 to obtain a deviation bound for the general case. We have:

$$\begin{aligned}
\left|x_k^{\mathcal{D}(\mathrm{SOS})}(t+1) - x_k^{\mathrm{SOS}}(t+1)\right| &= \left|\sum_{s=0}^{t}\sum_{\{i,j\}\in E}(Q_{k,i}(s)-Q_{k,j}(s))\cdot e_{i\to j}(t-s)\right| \\
&\leqslant \sum_{s=0}^{t}\sum_{\{i,j\}\in E}|Q_{k,i}(s)-Q_{k,j}(s)| \\
&\leqslant \sum_{s=0}^{t}\sum_{\{i,j\}\in E}\left(\left|Q_{k,i}(s)-\frac{s_k}{s}\cdot q(s)\right| + \left|Q_{k,j}(s)-\frac{s_k}{s}\cdot q(s)\right|\right) \\
&= d\cdot\sum_{s=0}^{t}\sum_{i=1}^{n}\left|Q_{k,i}(s)-\frac{s_k}{s}\cdot q(s)\right| \\
&\leqslant d\cdot\sqrt{n}\cdot\sum_{s=0}^{t}\left(\sum_{i=1}^{n}\left(Q_{k,i}(s)-\frac{s_k}{s}\cdot q(s)\right)^2\right)^{1/2} \qquad (5.22) \\
&\leqslant 4\cdot d\cdot\sqrt{n}\cdot\sqrt{2s_{\max}}\cdot\sum_{s=0}^{\infty}\left(\sqrt{\beta-1}\right)^s(s+1) \qquad (5.23) \\
&\leqslant 4\cdot d\cdot\sqrt{2ns_{\max}}\cdot\frac{1}{\left(1-\sqrt{\beta-1}\right)^2} \\
&\leqslant 16\cdot d\cdot\sqrt{2ns_{\max}}\cdot\frac{1}{1-\lambda}
\end{aligned}$$

where Equation (5.22) follows from the Cauchy-Schwarz inequality and Equation (5.23) follows from Lemma 5.9.(4).

□

5.2.2 Framework for Randomized SOS Schemes

In the next theorem we bound the deviation between continuous and discrete SOS using the randomized rounding scheme from Section 5.1.2. As mentioned earlier in this section, it is easy to see that the proof of Theorem 5.3 holds for the more general definitions for linearity and contribution of this section. Hence, we can state the following observation.

Observation 5.11. *In the setting of Section 5.2 the following holds for an arbitrary round t w.h.p.:* $\left|X_k^R(t) - x_k^{\mathcal{C}}(t)\right| = \mathcal{O}\left(\Upsilon^C(G) \cdot \sqrt{d \log n}\right)$.

Similar to Section 5.1, we can use Observation Observation 5.11 to show the next theorem:

Theorem 5.12. *Let $R = R(\mathrm{SOS})$ be a randomized-rounding discrete SOS process with optimal β obtained using our randomized rounding scheme. Then*

(1) $\Upsilon^{\mathrm{SOS}}(G) = \mathcal{O}\left(\frac{\sqrt{d}\cdot\log s_{\max}}{(1-\lambda)^{3/4}}\right)$.

(2) The deviation of R from the continuous SOS in an arbitrary round t is w.h.p.,

$$\left|X_k^R(t) - x_k^{\mathrm{SOS}}(t)\right| = \mathcal{O}\Big(\frac{d \cdot \log s_{\max} \cdot \sqrt{\log n}}{(1-\lambda)^{3/4}}\Big).$$

Proof outline. The bound on the refined local divergence is obtained using the formulation of Lemma 5.8 and the bound in Lemma 5.9. This bound together with the parametric deviation bound of Theorem 5.3 yield the second statement of the theorem.

Proof. We write

$$\begin{aligned}
\left(\Upsilon^{\mathrm{SOS}}(G)\right)^2 &= \sum_{t=0}^{\infty}\sum_{i=1}^{n} \max_{j\in N(i)} \left(Q_{k,i}(t) - Q_{k,j}(t)\right)^2 \\
&\leqslant \sum_{t=0}^{\infty}\sum_{i=1}^{n}\sum_{j\in N(i)} \left(Q_{k,i}(t) - Q_{k,j}(t)\right)^2 \\
&= \sum_{t=0}^{t_1-1}\sum_{i=1}^{n}\sum_{j\in N(i)} \left(Q_{k,i}(t) - Q_{k,j}(t)\right)^2 + \sum_{t=t_1}^{\infty}\sum_{i=1}^{n}\sum_{j\in N(i)} \left(Q_{k,i}(t) - Q_{k,j}(t)\right)^2
\end{aligned} \tag{5.24}$$

In the following, we use $\sigma := \beta - 1$ for brevity. Note that $0 < \sigma < 1$.

$$\sum_{t=0}^{t_1-1}\sum_{i=1}^{n}\sum_{j\in N(i)} \left(Q_{k,i}(t) - Q_{k,j}(t)\right)^2 \leqslant \sum_{t=0}^{t_1}\sum_{i=1}^{n}\sum_{j\in N(i)} 2\left((Q_{k,i}(t))^2 + (Q_{k,j}(t))^2\right)$$

$$\begin{aligned}
&\leqslant 4 \cdot d \cdot \sum_{t=0}^{t_1-1} \sum_{i=1}^{n} (Q_{k,i}(t))^2 \leqslant 4 \cdot d \cdot \sum_{t=0}^{t_1-1} \|Q(t)\,\widehat{\mathbf{k}}\|_2^2 \\
&= 4 \cdot d \cdot \sum_{t=0}^{t_1-1} \left(\|\widehat{\mathbf{k}}\|_2 \cdot \max_i \gamma_i(t) \right)^2 \\
&= 4 \cdot d \cdot \sum_{t=0}^{t_1-1} \left(\frac{1-\sigma^{t+1}}{1-\sigma} \right)^2 \\
&\leqslant 4 \cdot d \cdot t_1 \cdot \left(\frac{1-\sigma^{t_1}}{1-\sigma} \right)^2 \leqslant 4 \cdot d \cdot t_1 \cdot (1-\sigma)^{-2} \quad (5.25)
\end{aligned}$$

$$\begin{aligned}
&\sum_{t=t_1}^{\infty} \sum_{i=1}^{n} \sum_{j \in N(i)} (Q_{k,i}(t) - Q_{k,j}(t))^2 \\
&\leqslant \sum_{t=t_1}^{\infty} \sum_{i=1}^{n} \sum_{j \in N(i)} 2 \left(\left(Q_{k,i}(t) - \frac{s_k}{s} \cdot q(t) \right)^2 + \left(Q_{k,j}(t) - \frac{s_k}{s} \cdot q(t) \right)^2 \right) \\
&= 4 \cdot d \cdot \sum_{t=t_1}^{\infty} \sum_{i=1}^{n} \left(Q_{k,i}(t) - \frac{s_k}{s} \cdot q(t) \right)^2 \leqslant 8 \cdot d \cdot s_{\max} \cdot \sum_{t=t_1}^{\infty} \left(\sigma^t \, (t+1)^2 \right) \quad (5.26)
\end{aligned}$$

where the last inequality follows from part (4) of Lemma 5.9. The above summation can be bounded as follows,

$$\begin{aligned}
\sum_{t=t_1}^{\infty} \left(\sigma^t \, (t+1)^2 \right) &= \frac{d}{d\sigma} \left(\sigma \frac{d}{d\sigma} \left(\frac{\sigma^{t_1+1}}{1-\sigma} \right) \right) \\
&\leqslant \frac{2\sigma^{t_1+2}}{(1-\sigma)^3} + \frac{(2t_1+3) \cdot \sigma^{t_1+1}}{(1-\sigma)^2} + \frac{(t_1+1)^2 \cdot \sigma^{t_1}}{1-\sigma} \quad (5.27)
\end{aligned}$$

Let $t_1 = (\log s_{\max})/(1-\sigma)$. Note that $\sigma^{1/(1-\sigma)} \leqslant 1/e$. Then Equation (5.27) yields

$$\sum_{t=t_1}^{\infty} \left(\sigma^t \, (t+1)^2 \right) = \mathcal{O}\left(\frac{\log^2 s_{\max}}{s_{\max} \cdot (1-\sigma)^3} \right) \quad (5.28)$$

Combining Equations (5.28) and (5.26) and then Equation (5.25) we get

$$\left(\Upsilon^{\mathrm{SOS}}(G) \right)^2 = \mathcal{O}\left(\frac{d \cdot \log s_{\max}}{(1-\sigma)^3} \right) + \mathcal{O}\left(\frac{d \cdot \log^2 s_{\max}}{(1-\sigma)^3} \right) = \mathcal{O}\left(\frac{d \cdot \log^2 s_{\max}}{(1-\sigma)^3} \right)$$

Observe that $1-\sigma = (1-\sqrt{\sigma})(1+\sqrt{\sigma}) \geqslant (1-\sqrt{\sigma}) = 1-\sqrt{\beta-1}$, where we have

$$1-\sqrt{\beta-1} = 1 - \frac{\lambda}{1+\sqrt{1-\lambda^2}} = \frac{1-\lambda+\sqrt{1-\lambda^2}}{1+\sqrt{1-\lambda^2}} \geqslant \frac{1}{2} \cdot \left(1-\lambda+\sqrt{1-\lambda^2} \right)$$

$$\geqslant \frac{1}{2} \cdot \sqrt{1-\lambda} \cdot \left(\sqrt{1-\lambda} + \sqrt{1+\lambda} \right) \geqslant \frac{1}{2} \cdot \sqrt{1-\lambda}$$

Therefore,

$$\Upsilon^{\mathrm{SOS}}(G) = \mathcal{O}\left(\frac{\sqrt{d} \cdot \log s_{\max}}{(1-\lambda)^{3/4}} \right).$$

This finishes the proof of the first statement. The bound in the second statement follows immediately from statement (1) and Theorem 5.3. □

5.2.3 Experimental Simulations

The second-order process was studied only in a few publications before. To the best of our knowledge, our work is the first to analyze a randomized second-order process. Absence of a preset baseline for comparison of the results lead us to perform simulations and compare our analytic bounds with those suggested by experiments.

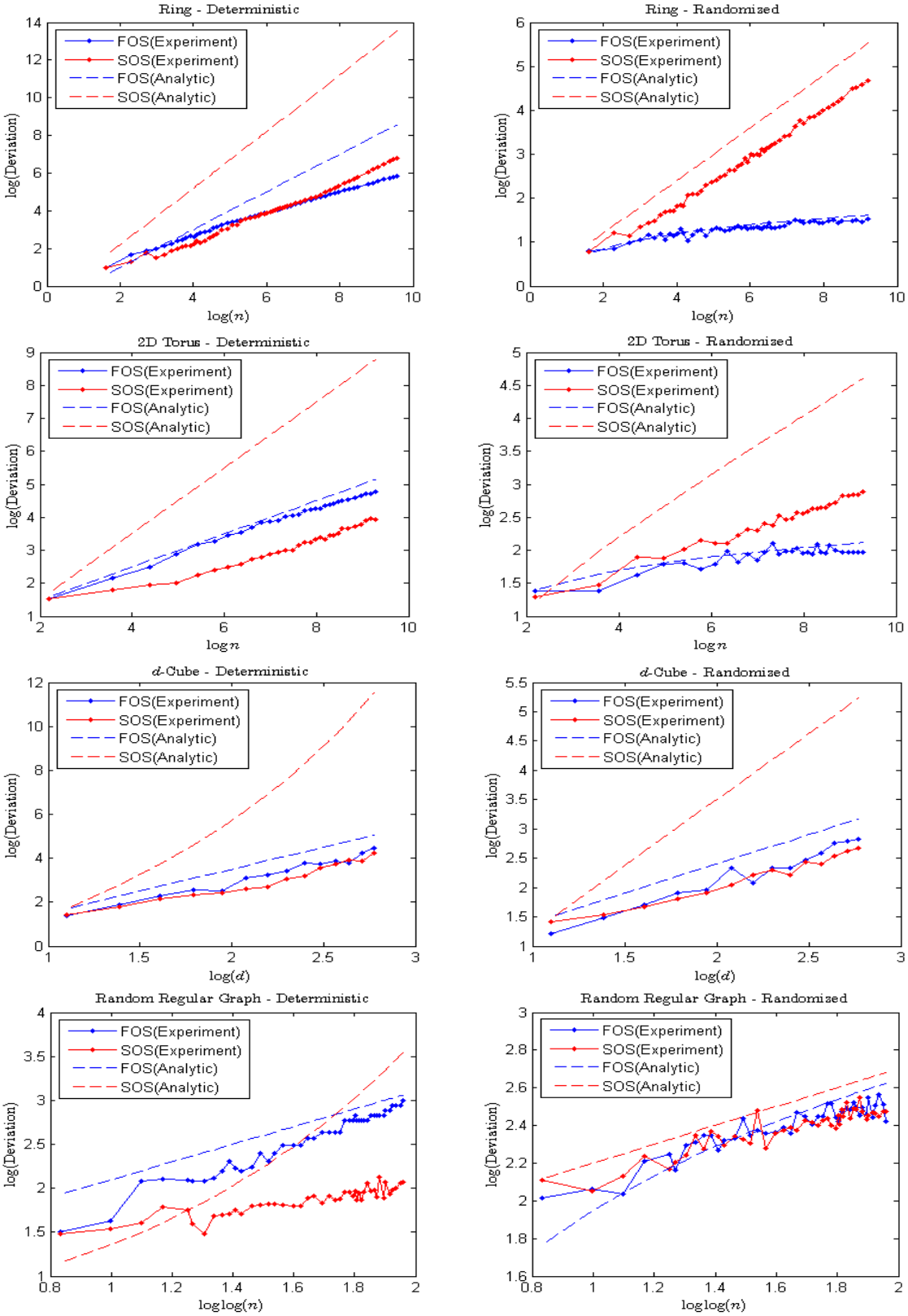
Figure 5.1: For the purpose of comparison, we simulated SOS and FOS discrete processes in the homogeneous model (both randomized and deterministic versions) for several graph classes. We measured the maximum deviation of each discrete process from its continuous counterpart ($\max_{i,t} |x_i^C(t) - x_i^D(t)|$) over tens of runs. The diagrams show how this deviation varies with n, the order of the graph in each graph class.

The empirical results for the deterministic SOS are polynomially better than the current analytic bound for all the graphs considered. This suggests that our deterministic bound is far from being tight. On the other hand, both in the randomized and deterministic case, there are cases where FOS has clearly better performance compared to SOS (rings, expanders (deterministic case), and tori (randomized case)). This means that it might not be possible to derive SOS bounds that are in general better than FOS bounds. However, in some cases SOS performs very similar to FOS in practice (hypercubes, and expanders(randomized case)).

5.3 Negative Load for SOS Schemes

In the second-order diffusion nodes might not have enough load to satisfy all their neighbours' demand. This situation, which we refer to as *negative load*, motivates studying by how much a node's load may become negative. Here we study the minimum amount of load that nodes need in order to prevent this event.

Figure 5.1: Comparison of discrete FOS and SOS processes on different network topologies in terms of the maximum deviation from their continuous counterpart processes. The diagrams show the growth of deviation with respect to n.



In the following we calculate a bound on the minimum load of every node that holds during the whole balancing process. Note that, if every processor has minimum load at the beginning of the balancing process, there will be no processor with negative load. Hence, these bounds can also be regarded as bounds on the minimum load of every processor in order to avoid negative load.

Let $\bar{x} = (\bar{x}_1, \ldots, \bar{x}_n)$ be the balanced load vector. Define $\Delta(t) := \|x(t) - \bar{x}\|_\infty$, and $\Phi(t) := \|x(t) - \bar{x}\|_2$, where $\|.\|$ is the norm operator. Then the following observation estimates the load at the end of every step.

Observation 5.13. *In continuous SOS with $\beta = \beta_{opt}$ we have $x(t) \geqslant -\sqrt{n} \cdot \Delta(0)$.*

Proof. We first note that

$$\Delta(t) \leqslant \Phi(t) \overset{(*)}{\leqslant} \lambda^t \cdot \Phi(0) \leqslant \lambda^t \cdot \sqrt{n} \cdot \Delta(0), \tag{5.29}$$

where $(*)$ follows from a result by Muthukrishnan et al. [57]. They show that $x(t) = M(t)\, x(0)$ for an $n \times n$ matrix $M(t)$ defined recursively. They also show that $\Phi(t) \leqslant \gamma(t) \cdot \Phi(0)$ where $\gamma(t)$ is the second largest eigenvalue in magnitude of $M(t)$ and $\gamma(t) \leqslant \lambda^t$ [57, Proof of Theorem 2]. Though they only consider homogeneous networks, their argument also applies to the heterogeneous case. The proof now follows by considering the facts $-x(t) \leqslant \Delta(t)$ and $\lambda^t < 1$. □

It should be noted that load during a balancing step can be lower than the bound in Observation 5.13 since Observation 5.13 considers only snapshots of the network at the end of each round. However, it might be possible that a node has to send more load items to some of its neighbours than it has at the beginning of round t, but its load is still positive at the end of round t. This can happen if it also receives many load items from other neighbours in round t. To study the negative load issue it is helpful to divide every round in two distinct steps; in the first step, all the nodes send out their outgoing flows. In the second step, they receive incoming flows forwarded by their neighbours in the first step. At the end of the first step all the outgoing flows are sent out but no incoming flow is yet received. To prevent negative load the load of every node has to be non-negative at this point. We refer to this point as the *transient state*, and use $\breve{x}_i(t)$ to denote the load in the transient state of round t. Note that we always have $\breve{x}_i(t) \leqslant x_i(t)$ and $\breve{x}_i(t) \leqslant x_i(t+1)$. The following theorem provides a lower bound on $\breve{x}_i(t)$.

Theorem 5.14. *In a continuous SOS process with $\beta = \beta_{opt}$ we have*

$$\breve{x}_i(t) \geqslant -\mathcal{O}\left(\frac{\sqrt{n}\cdot\Delta(0)}{\sqrt{1-\lambda}}\right).$$

Proof. Observe that for $t > 1$ and an arbitrary node i we have

$$y_{i,j}(t) = (\beta-1)\cdot y_{i,j}(t-1) + \beta\cdot\alpha_{i,j}\cdot\left(\frac{x_i(t)}{s_i} - \frac{x_j(t)}{s_j}\right) \tag{5.30}$$

Thus we have (Note that $\bar{x}_i/s_i = \bar{x}_j/s_j$),

$$\begin{aligned}
&\sum_{j\in N(i)} |y_{i,j}(t)| \\
&\leqslant (\beta-1)\cdot\sum_{j\in N(i)} |y_{i,j}(t-1)| + \beta\cdot\sum_{j\in N(i)}\alpha_{i,j}\cdot\left|\frac{x_i(t)}{s_i} - \frac{x_j(t)}{s_j}\right| \\
&\leqslant (\beta-1)\cdot\sum_{j\in N(i)} |y_{i,j}(t-1)| + \beta\cdot\sum_{j\in N(i)}\alpha_{i,j}\cdot\left(\left|\frac{x_i(t)}{s_i} - \frac{\bar{x}_i}{s_i}\right| + \left|\frac{x_j(t)}{s_j} - \frac{\bar{x}_j}{s_j}\right|\right)
\end{aligned}$$

Let $g(t) := \sum_{j\in N(i)} |y_{i,j}(t)|$. Recall that $\beta < 2$, and for all i, $s_i \geqslant 1$ and $\sum_{j\in N(i)\cup\{i\}}\alpha_{i,j} = 1$. So we get

$$\begin{aligned}
g(t+1) &\leqslant (\beta-1)\cdot g(t) + 2\cdot\sum_{j\in N(i)}\alpha_{i,j}\cdot\left(|x_i(t+1)-\bar{x}_i| + \left|x_j(t+1)-\bar{x}_j\right|\right) \\
&\leqslant (\beta-1)\cdot g(t) + 4\cdot\Delta(t+1)\cdot\sum_{j\in N(i)}\alpha_{i,j} \\
&\leqslant (\beta-1)\cdot g(t) + 4\cdot\Delta(t+1) \\
&\leqslant (\beta-1)\cdot g(t) + 4\cdot\lambda^{t+1}\cdot\sqrt{n}\cdot\Delta(0) \qquad \text{(By Equation (5.29))}
\end{aligned} \tag{5.31}$$

Also note that $g(0) < \Delta(0)$. From the recurrence of Equation (5.31) we get:

$$\begin{aligned}
g(t+1) &\leqslant 4\sum_{k=0}^{t}(\beta-1)^{t-i}\cdot\lambda^i\cdot\sqrt{n}\cdot\Delta(0) \\
&= 4\sqrt{n}\cdot\Delta(0)\cdot\frac{\lambda^{t+1}-(\beta-1)^{t+1}}{\lambda-(\beta-1)} \\
&\leqslant 4\sqrt{n}\cdot\Delta(0)\cdot\frac{\lambda}{\lambda-(\beta-1)}
\end{aligned} \tag{5.32}$$

where the last inequality holds because $\lambda < 1$. On the other hand, we have

$$\lambda-(\beta-1) = \frac{(1+\lambda)\sqrt{1-\lambda^2}-(1-\lambda)}{1+\sqrt{1-\lambda^2}} > \frac{\sqrt{1-\lambda}\cdot\lambda}{4}$$

Therefore, we can apply the above to Equation (5.32) to get: $g(t) = \mathcal{O}\left(\sqrt{n} \cdot \Delta(0)/\sqrt{1-\lambda}\right)$

To complete the proof, we note that $\breve{x}_i(t) \geqslant x_i(t) - g(t)$, while by Observation 5.13 we have $x_i(t) \geqslant -\sqrt{n} \cdot \Delta(0)$. This yields the desired lower bound of $-\mathcal{O}\left(\frac{\sqrt{n} \cdot \Delta(0)}{\sqrt{1-\lambda}}\right)$. □

Remark 5.15. *We can apply a similar argument as in the proof of Theorem 5.14 to get a lower bound for the randomized discrete second-order process ($R = R(SOS)$) in many cases. For instance, if $s_{\max}$ is polynomial in n and $d/(1-\lambda)^{3/4} = \mathcal{O}(n^{0.5-\varepsilon})$ for some $\varepsilon > 0$[1] then the asymptotic lower bound obtained in Observation 5.13 also holds for R. To show this, we rewrite Equation* (5.30) *as follows:*

$$Y_{i,j}(t) \leqslant (\beta - 1) \cdot Y_{i,j}(t-1) + \beta \cdot \alpha_{i,j} \cdot \left(\frac{X_i(t)}{s_i} - \frac{X_j(t)}{s_j}\right) + d,$$

which results in the recursion

$$G(t+1) \leqslant (\beta - 1) \cdot G(t) + 4 \cdot \lambda^{t+1} \cdot \sqrt{n} \cdot \Delta(0) + d^2,$$

Proceeding with similar steps as in the proof of Theorem 5.14 we get the following theorem:

Theorem 5.16. *Under the conditions of Remark 5.15, in a randomized discrete SOS process $R = R(\mathrm{SOS})$ with $\beta = \beta_{opt}$ we have*

$$\breve{X}_i^R(t) \geqslant -\mathcal{O}\left(\frac{\sqrt{n} \cdot \Delta(0) + d^2}{\sqrt{1-\lambda}}\right).$$

5.3.1 Experimental Simulations

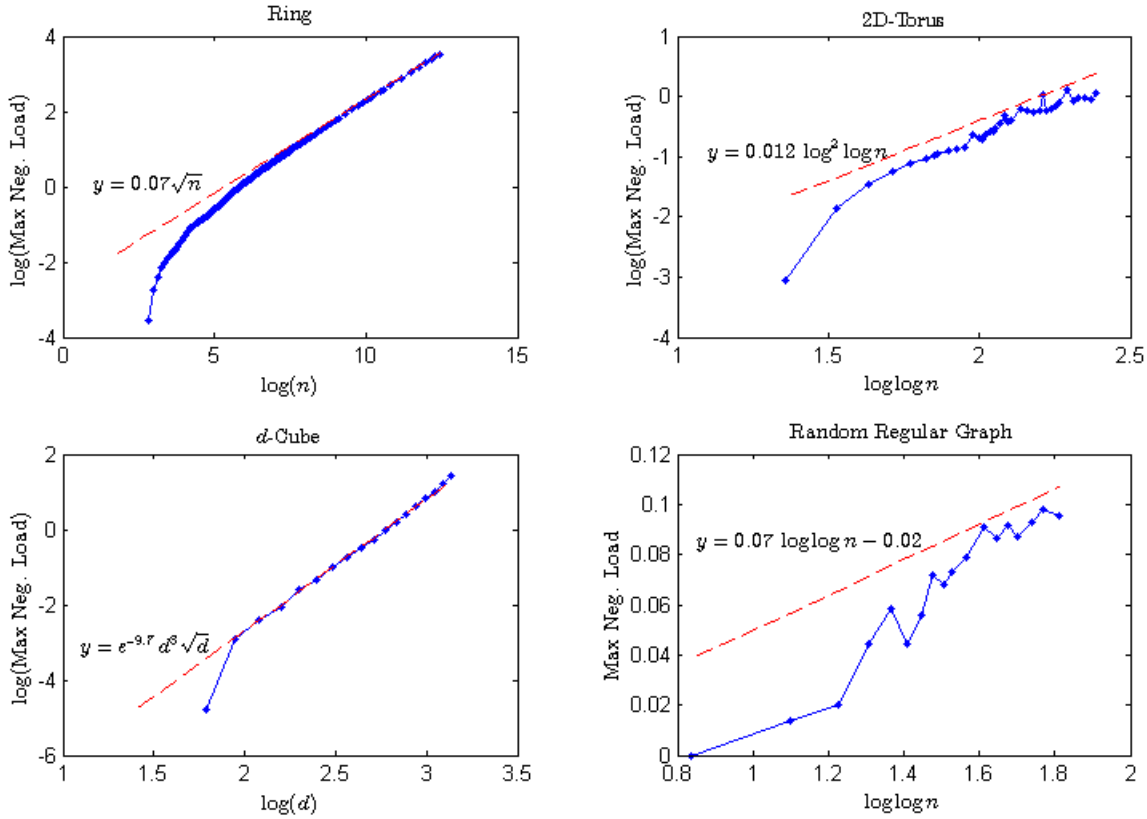
To investigate the negative load situations, we ran simulations for different graph classes and load distributions. In our experiments, $\max_i x_i(0) = 1$ and $\min_i x_i(0) = 0$. For each network topology, we measured both $\max_{i,t}(-x_i(t))$ and $\max_{i,t}(-\breve{x}_i(t))$ over tens of runs with different initial load vectors (both randomly and deterministically generated vectors). Surprisingly, for each fixed network graph, the maximum negative load over different runs was always equal to the maximum negative transient load over those runs. Hence, each of the diagrams in Figure 5.2 show only one data series. By their construction, our experiments guarantee to obtain the exact value for $\max_{i,t}(-x_i(t))$.

[1] *This is true, e.g., for tori with four or more dimensions, hypercubes, and expanders.*

To do this, we considered the symmetric matrix $M(t)$ where $x(t) = M(t)x(0)$ and found i, t such that $\sum_{j:M_{i,j}(t)<0}(-M_{i,j}(t))$ is maximized. To construct the instance that produces the most negative $x_i(t)$, it suffices to put a load of 1 on $j : M_{i,j}(t) < 0$ and zero elsewhere. Unfortunately, this method works only for $\max_{i,t}(-x_i(t))$, and we were not able to verify if our experiments captured the worst state in terms of the negative *transient* load ($\max_{i,t}(-\breve{x}_i(t))$).

Our experiments suggest that the bound in Observation 5.13 is asymptotically tight for rings. As can be seen in Figure 5.2, for large n the experiment data lie almost in parallel to the curve $y = \Theta(\sqrt{n})$. However, for other graph classes such as hypercubes, the bound in Observation 5.13 appears to be too loose compared to what suggested by the experiment data.

Figure 5.2: Simulation results for maximum (transient) negative load (in absolute value). The diagrams show how the negative load grows with n in different graph classes.



Appendix A

Tools

Lemma A.1 ([44, Fact 2])**.** *From the Courant-Fischer Minimax Theorem it follows that*

$$\lambda_2 = \min\left(\frac{\mathbf{x}^T L\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \;\middle|\; \mathbf{x} \perp \mathbf{u}, \mathbf{x} \neq \mathbf{0}\right)$$

where $\mathbf{u} = (1, 1, \ldots, 1)^{\mathrm{T}}$ *is the eigenvector corresponding to* $\lambda_1 = 0$ *and* $\mathbf{x} \perp \mathbf{u}$ *denotes* $\mathbf{x}$ *is orthogonal to* $\mathbf{u}$*.*

Lemma A.2 (Hoeffding's bound [47, Theorem 2])**.** *Suppose* $s_n = X_1 \ldots X_n$ *where the* X_i*'s are independent random variables, and for each* $i = 1, \ldots, n, X_i \in [a_i, b_i]$*. Then, for any* $\epsilon > 0$*,*

$$\mathbf{Pr}\left[s_n - \mathbf{Ex}\left[s_n\right] > \epsilon\right] \leqslant \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

Lemma A.3 (Hoeffding's Bound: Adaptation for Randomized Rounding)**.** *Let* $X_1, \ldots, X_k$ *be* k *independent random variables, and* $p_1, \ldots, p_k$ *be constants where for all* $i, 0 < |p_i| < 1$*. Suppose* X_i *is* $p_i - 1$ *with probability* p_i *and* p_i *otherwise. If we define the random variable* $X = \sum_{1 \leqslant i \leqslant k} X_i$*, then we have for any* $\delta > 0$ *that*

$$\mathbf{Pr}\left[\,|X| \geqslant \delta\,\right] \leqslant 2\exp\left(-2\delta^2/k\right).$$

Proof. We first note that for each i, we have

$$\mathbf{Ex}\left[X_i\right] = (p_i - 1) \cdot p_i + p_i \cdot (1 - p_i) = 0;$$

Hence, by the linearity of expectation we get $\mathbf{Ex}\left[X\right] = \sum_{1 \leqslant i \leqslant k} \mathbf{Ex}\left[X_i\right]$. Also, for each i, $p_i - 1 \leqslant X_i \leqslant p_i$. Therefore, we can apply the Hoeffding's bound [47] to get $\mathbf{Pr}\left[\,|X| \geqslant \delta\,\right] \leqslant 2\exp\left(-2\delta^2/k\right)$, as required. □

Theorem A.4 (Method of Averaged Bounded Differences [26, p. 68]). *Let $U_1, \ldots, U_n$ be an arbitrary set of random variables and let f be a function of these random variables satisfying the property that for each $1 \leqslant \ell \leqslant n$, there is a non-negative c_ℓ such that*

$$\left| \mathbf{Ex}\left[f \mid U_\ell, U_{\ell-1}, \ldots, U_1\right] - \mathbf{Ex}\left[f \mid U_{\ell-1}, \ldots, U_1\right] \right| \leqslant c_\ell.$$

Then for any $\delta > 0$,

$$\mathbf{Pr}\left[|f - \mathbf{Ex}\left[f\right]| > \delta\right] \leqslant 2 \exp\left(-\frac{\delta^2}{2c}\right),$$

where $c := \sum_{\ell=1}^{n} c_\ell^2$.

Theorem A.5 ([41, Theorem 4.2]). *For all initial load vectors on the d-dimensional hypercube with n nodes, the deviation between the idealized process and a discrete process with accumulated rounding errors at most ξ is $O(\xi \log^{3/2} n)$ at all times.*

Theorem A.6 ([41, Theorem 5.4]). *For all initial load vectors on the d-dimensional torus with n nodes, the deviation between the idealized process and a discrete process with accumulated rounding errors at most ξ is $O(\xi)$ at all times.*

Lemma A.7 ([57, Lemma 1]). *Consider $\mathbf{w}^{t+1} = M\mathbf{w}^t$ where M is a diffusion matrix. We have $\Phi_t \leqslant \lambda^{2t}\Phi_0$ (where λ is the second largest eigenvalue in absolute value of M and $\Phi_t := \sum_i (\mathbf{w}_i^t - \bar{w})^2$, where $\bar{w} = \sum_i \mathbf{w}_i^t/n$)*

Lemma A.8. *For an arbitrary $1 \leqslant k \leqslant n$, let the vector $\mathbf{a}$ be such that $\mathbf{a}_i := M_{k,i}^t - \frac{s_k}{s}$. Then we have:*

$$\|\mathbf{a}\|_2^2 \leqslant 2\, s_{\max}\, \lambda^{2t}$$

Proof. Let $\widehat{\mathbf{a}} := \widehat{\mathbf{k}} - \frac{s_k}{s} \cdot \mathbf{1}_n$. Note that $\mathbf{a} = \widehat{\mathbf{a}}\, M^t$. Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the eigenvectors of M^t with eigenvalues $\lambda_1^t, \ldots, \lambda_n^t$, and λ^t be the second largest eigenvalue. Using the fact that $M = I - LS^{-1}$, it is not hard to see that $S^{-1}M^t$ is symmetric. Hence, for each right eigenvector $\mathbf{v}_i$ of M^t there is a left eigenvector $\mathbf{u}_i = S^{-1}\mathbf{v}_i$ with the same eigenvalue λ_i^t as proved in the following:

$$(M^t)^T\mathbf{u}_i = (M^t)^T S^{-1} S\mathbf{u}_i = (S^{-1}M^t)^T\mathbf{v}_i = S^{-1}M^t\mathbf{v}_i = \lambda_i^t S^{-1}\mathbf{v}_i = \lambda_i^t\mathbf{u}_i$$

Also, note that $S^{-1}M^tS\mathbf{u}_i = S^{-1}M^t\mathbf{v}_i = \lambda_i^t S^{-1}\mathbf{v}_i = \lambda_i^t\mathbf{u}_i$. As a result, $\mathbf{u}_i$'s are eigenvectors of $S^{-1}M^tS$, which is symmetric because it is the product of symmetric matrices $S^{-1}M^t$ and S. Therefore, $\mathbf{u}_1, \ldots, \mathbf{u}_n$ form an orthonormal basis; so we can write

$\widehat{\mathbf{a}} = \sum_{i=1}^{n} c_i \mathbf{u}_i$. Now we write

$$\mathbf{a} = \widehat{\mathbf{a}}\, M^t = \sum_{i=1}^{n} c_i \mathbf{u}_i M^t = \sum_{i=1}^{n} \lambda_i^t c_i \mathbf{u}_i$$

Therefore,

$$\|\mathbf{a}\|_2^2 = \sum_{i=1}^{n} \lambda_i^{2t} c_i^2 \|\mathbf{u}_i\|_2^2 \leqslant \lambda^{2t} \sum_{i=1}^{n} c_i^2 \|\mathbf{u}_i\|_2^2 = \lambda^{2t} \|\widehat{\mathbf{a}}\|_2^2 \tag{A.1}$$

where the inequality uses the fact that $\widehat{\mathbf{a}} \perp (s_1, \ldots, s_n)$ which is the eigenvector corresponding to the largest eigenvalue. Also, the last equality follows from the fact that $\mathbf{u}_i$'s form an orthonormal basis. On the other hand,

$$\|\widehat{\mathbf{a}}\|_2^2 \leqslant n \cdot \frac{s_k^2}{s^2} + 1 \leqslant \frac{n\, s_k^2}{(n-1+s_k)^2} + 1 \leqslant \frac{n\, s_k^2}{2(n-1)s_k} + 1 \leqslant s_k + 1 \leqslant 2 s_{\max}$$

Together with Equation (A.1), this yields

$$\|\mathbf{a}\|_2^2 \leqslant 2\, s_{\max} \lambda^{2t},$$

as required. □

A.1 Hypercube Facts

Lemma A.9 ([10, Lemma D.1])**.** *For the d-dimensional hypercube with $n = 2^d$ vertices the following statements hold.*

(1) For any edge $(i,j) \in E$, any node $k \in V$ and any time-step $\vartheta \in \mathbb{N}$,

$$\sum_{t=\vartheta}^{\infty} \left| \mathbf{P}_{i,k}^t - \mathbf{P}_{j,k}^t \right| = \left| \sum_{t=\vartheta}^{\infty} \left(\mathbf{P}_{i,k}^t - \mathbf{P}_{j,k}^t \right) \right|.$$

(2) Let 0 also denote the node $0^{\log n} \in V$. For any two vertices i, j with $(i,j) \in E$, $\|i\|_1 = p$ and $\|j\|_1 = p+1$ we have

$$\sum_{t=0}^{\infty} \left(\mathbf{P}_{i,0}^t - \mathbf{P}_{j,0}^t \right) = \frac{1}{n} \cdot \frac{\log n + 1}{\binom{d}{p} (\log n - p)} \cdot \sum_{\ell=p+1}^{\log n} \binom{d}{\ell}.$$

Proof. (1) As shown in [23, Lemma 6], it holds for all triples of vertices i, j, k with $\operatorname{dist}(i,k) \leqslant \operatorname{dist}(j,k)$ that for all time steps $t \in \mathbb{N}$, $\mathbf{P}_{i,k}^t \geqslant \mathbf{P}_{j,k}^t$. This immediately implies that

$$\sum_{t=\vartheta}^{\infty} \left| \mathbf{P}_{i,k}^t - \mathbf{P}_{j,k}^t \right| = \left| \sum_{t=\vartheta}^{\infty} \left(\mathbf{P}_{i,\ell}^t - \mathbf{P}_{j,k}^t \right) \right|.$$

(2) The second claim is a slight reformulation of [55, Thm. 5]. Note that in these statements we have $\delta = 1$ and $w_u - w_v = \alpha\, n\, \delta \left(\sum_{t=0}^{\infty} \mathbf{M}^t_{0,i} - \mathbf{M}^t_{0,j} \right)$ with $\alpha = \frac{1}{\log n + 1}$ ($\mathbf{M}$ is the same matrix as $\mathbf{P}$).

□ □

Lemma A.10 ([23, Lemma 6])**.** *For any fixed t, $\mathbf{P}^t_{p,0}$ is decreasing for $0 \leqslant p \leqslant \log n$.*

Lemma A.11 ([10, Lemma D.3])**.** *For any $t \in \mathbb{N}_{>0}$, $\mathbf{P}^t_{0,0} \leqslant \frac{1}{\log n + 1}$. Moreover, for any $1 \leqslant p \leqslant (\log n)/2$ and $t \in \mathbb{N}$,*

$$\mathbf{P}^t_{p,0} \leqslant \frac{1}{\binom{\log n}{p}},$$

and for any $\log n/2 \leqslant p \leqslant \log n$,

$$\mathbf{P}^t_{p,0} \leqslant \frac{1}{\binom{\log n}{\log n/2}}.$$

Bibliography

[1] H. Ackermann, H. Röglin, and B. Vöcking. On the impact of combinatorial structure on congestion games. *J. ACM*, 55(6):25:1–25:22, 2008.

[2] C. P. J. Adolphs and P. Berenbrink. Improved bounds for discrete diffusive load balancing. In *IPDPS*, pages 820–826. IEEE Computer Society, 2012.

[3] C. P.J. Adolphs and P. Berenbrink. Distributed selfish load balancing with weights and speeds. In *PODC*, pages 135–144. ACM, 2012.

[4] H. Akbari and P. Berenbrink. Parallel rotor walks on finite graphs and applications in discrete load balancing. In *SPAA*, page to appear. ACM, 2013.

[5] H. Akbari, P. Berenbrink, and T. Sauerwald. A simple approach for adapting continuous load balancing processes to discrete settings. In *PODC*, pages 271–280, 2012.

[6] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.

[7] E. Bampas, L. Gasieniec, R. Klasing, A. Kosowski, and T. Radzik. Robustness of the rotor-router mechanism. In *OPODIS*, pages 345–358, 2009.

[8] P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. Goldberg, Z. Hu, and R. Martin. Distributed selfish load balancing. In *SODA*, pages 354–363. ACM, 2006.

[9] P. Berenbrink, T. Friedetzky, and Z. Hu. A new analytical method for parallel, diffusion-type load balancing. *J. Parallel Distrib. Comput.*, 69(1):54–61, 2009.

[10] P. Berenbrink, C. Cooper, T. Friedetzky, T. Friedrich, and T. Sauerwald. Randomized diffusion for indivisible loads. In *SODA*, pages 429–439. SIAM, 2011.

[11] P. Berenbrink, T. Friedetzky, I.n Hajirasouliha, and Z. Hu. Convergence to equilibria in distributed, selfish reallocation processes with weighted tasks. *Algorithmica*, 62(3–4):767–786, 2012.

[12] P. Berenbrink, M. Hoefer, and T. Sauerwald. Distributed selfish load balancing on networks. *ACM Trans. Algorithms*, 11(1):2:1–2:29, 2014.

[13] S. Bhatt, S. Even, D. Greenberg, and R. Tayar. Traversing directed eulerian mazes. *Journal of Graph Algortihms and Applications*, 6(2):157–173, 2002.

[14] J. E. Boillat. Load balancing and poisson equation in a graph. *Concurrency and Computation: Practice and Experience*, 2:289–314, 1990.

[15] S. Chien and A. Sinclair. Convergence to approximate nash equilibria in congestion games. In *SODA*, pages 169–178. SIAM, 2007.

[16] C. Cooper, D. Ilcinkas, R. Klasing, and A. Kosowski. Derandomizing random walks in undirected graphs using locally fair exploration strategies. *Distributed Computing*, 24(2):91–99, 2011.

[17] J. Cooper, B. Doerr, J. Spencer, and G. Tardos. Deterministic random walks on the integers. volume 28, pages 2072–2090. Academic Press Ltd., 2007.

[18] J. N. Cooper and J. Spencer. Simulating a random walk with constant error. *Comb. Probab. Comput.*, 15:815–822, 2006.

[19] J. N. Cooper, B. Doerr, T. Friedrich, and J. Spencer. Deterministic random walks on regular trees. In *SODA*, pages 766–772, 2008.

[20] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.*, 7:279–301, 1989.

[21] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. *ACM Trans. Algorithms*, 3(1):4:1–4:17, 2007.

[22] A. Czumaj, C. Riley, and C. Scheideler. Perfectly balanced allocation. In *RANDOM-APPROX*, volume 2764 of *Lecture Notes in Computer Science*, pages 240–251. Springer, 2003.

[23] P. Diaconis, R. L. Graham, and J. A. Morrison. Asymptotic analysis of a random walk on a hypercube with many dimensions. *Random Structures Algorithms*, 1(1): 51–72, 1990.

[24] B. Doerr and T. Friedrich. Deterministic random walks on the two-dimensional grid. *Comb. Probab. Comput.*, 18(1-2):123–144, 2009.

[25] D. Dubhashi and D. Ranjan. Balls and bins: A study in negative dependence. *Random Structures & Algorithms*, 13:99–124, 1996.

[26] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[27] R. Elsässer and B. Monien. Load balancing of unit size tokens and expansion properties of graphs. In *SPAA*, pages 266–273, 2003.

[28] R. Elsässer and T. Sauerwald. Discrete load balancing is (almost) as easy as continuous load balancing. In *PODC*, pages 346–354, 2010.

[29] R. Elsässer, B. Monien, and R. Preis. Diffusion schemes for load balancing on heterogeneous networks. *Theory Comput. Syst.*, 35(3):305–320, 2002.

[30] R. Elsässer, B. Monien, and S. Schamberger. Distributing unit size workload packages in heterogeneous networks. *J. Graph Algorithms Appl.*, 10(1):51–68, 2006.

[31] L. Epstein. Equilibria for two parallel links: the strong price of anarchy versus the price of anarchy. *Acta Inf.*, 47(7-8):375–389, 2010.

[32] L. Epstein and R. van Stee. The price of anarchy on uniformly related machines revisited. *Information and Computation*, 212:37–54, 2012.

[33] E. Even-Dar and Y. Mansour. Fast convergence of selfish rerouting. In *SODA*, pages 772–781. SIAM, 2005.

[34] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to nash equilibrium in load balancing. *ACM Trans. Algorithms*, 3(3), 2007.

[35] A. Fabrikant, C. Papadimitriou, and K. Talwar. The complexity of pure nash equilibria. In *STOC*, pages 604–612. ACM, 2004.

[36] R. Feldmann, M. Gairing, T. Lücking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *ICALP*, pages 514–526. Springer-Verlag, 2003.

[37] G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, 19:312–320, 1979.

[38] D. Fotakis, S. C. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. G. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. *Theor. Comput. Sci.*, 410(36):3305–3326, 2009.

[39] T. Friedrich and T. Sauerwald. Near-perfect load balancing by randomized rounding. In *STOC*, pages 121–130, 2009.

[40] T. Friedrich and T. Sauerwald. The cover time of deterministic random walks. *Electr. J. Comb.*, 17(1), 2010.

[41] T. Friedrich, M. Gairing, and T. Sauerwald. Quasirandom load balancing. *SIAM J. Comput.*, 41(4):747–771, 2012.

[42] M. Gairing, T. Lücking, M. Mavronicolas, B. Monien, and P. Spirakis. Structure and complexity of extreme nash equilibria. *Theor. Comput. Sci.*, 343(1-2):133–157, 2005.

[43] L. Gasieniec and T. Radzik. Memory efficient anonymous graph exploration. In *WG*, pages 14–29, 2008.

[44] B. Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. Comput. Syst. Sci.*, 53:357–370, 1996.

[45] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. W. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM J. Comput.*, 29(1):29–64, 1999.

[46] P. W. Goldberg. Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In *PODC*, pages 131–140. ACM, 2004.

[47] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Parallel Distrib. Comput.*, 58(301):13–30, 1963.

[48] A. E. Holroyd and J. Propp. Rotor walks and markov chains. *Algorithmic Probability and Combinatorics*, pages 105–126, 2010.

[49] S. H. Hosseini, B. E. Litow, M. I. Malkawi, and K. Vairavan. Distributed algorithms for load balancing in very large homogeneous systems. In *Proc. of the 1987 Fall Joint Computer Conference on Exploring technology: today and tomorrow*, ACM '87, pages 397–404. IEEE Computer Society, 1987.

[50] S. H. Hosseini, B. Litow, M. Malkawi, J. McPherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *J. Parallel Distrib. Comput.*, 10(2):160–166, 1990.

[51] S. Ieong, R. McGrew, E. Nudelman, Y. Shoham, and Q. Sun. Fast and compact: a simple class of congestion games. In *AAAI*, pages 489–494. AAAI Press, 2005.

[52] S. Kijima, K. Koga, and K. Makino. Deterministic random walks on finite graphs. In *ANALCO*, pages 16–25, 2012.

[53] R. Klasing, A. Kosowski, D. Pająk, and T. Sauerwald. The multi-agent rotor-router on the ring: A deterministic alternative to parallel random walks. In *PODC*, pages 365–374. ACM, 2013.

[54] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *STACS*, pages 404–413. Springer-Verlag, 1999.

[55] H. Meyerhenke and T. Sauerwald. Analyzing disturbed diffusion on networks. In *Proceedings of the 17th international conference on Algorithms and Computation*, 2006.

[56] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[57] S. Muthukrishnan, B. Ghosh, and M. H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory Comput. Syst.*, 31 (4):331–354, 1998.

[58] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *STOC*, pages 581–592. ACM, 1992.

[59] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.

[60] V. B. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy. Eulerian walkers as a model of self-organised criticality. Technical Report cond-mat/9611019. TIFR-TH-96-43, Tata Inst. Fundam. Res., 1996.

[61] M. Raab and A. Steger. "balls into bins" - a simple and tight analysis. In *RANDOM-APPROX*, pages 159–170. Springer-Verlag, 1998.

[62] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load-balancing schemes. In *FOCS*, pages 694–703. IEEE Computer Society, 1998.

[63] R. W. Rosenthal. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[64] T. Sauerwald and H. Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *FOCS*, pages 341–350. IEEE Computer Society, 2012.

[65] P. Schuurman and T. Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS J. on Computing*, 19(1):52–63, 2007.

[66] R. Subramanian and I. D. Scherson. An analysis of diffusive load-balancing. In *SPAA*, pages 220–225. ACM, 1994.

[67] B. Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, 2003.

[68] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Smell as a computational resource - a lesson we can learn from the ant. In *Proc. ISTCS'96 - 4th Israel Symposium on the Theory of Computing and Systems*, pages 219–230, 1996.

[69] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.

[70] P. Wan. Near-optimal conflict-free channel set assignments for an optical cluster-based hypercube network. *J. Comb. Optim.*, 1(2):179–186, 1997.

[71] V. Yanovski, I. A. Wagner, and A. M. Bruckstein. A distributed ant algorithm for efficiently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.