Improved Analysis of Deterministic Load-Balancing Schemes

Petra Berenbrink, Ralf Klasing, Adrian Kosowski, Frederik Mallmann-Trenn, Przemyslaw Uznanski

▶ **To cite this version:**

Petra Berenbrink, Ralf Klasing, Adrian Kosowski, Frederik Mallmann-Trenn, Przemyslaw Uznanski. Improved Analysis of Deterministic Load-Balancing Schemes. 2014. hal-00979691v3

HAL Id: hal-00979691

https://inria.hal.science/hal-00979691v3

Preprint submitted on 19 Jul 2014 (v3), last revised 22 Feb 2015 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Improved Analysis of Deterministic Load-Balancing Schemes

Petra Berenbrink[1], Ralf Klasing[2], Adrian Kosowski[3], Frederik Mallmann-Trenn[1], and Przemysław Uznański[4]

[1]Simon Fraser University, {petra, fmallman}@sfu.ca
[2]CNRS – LaBRI – Université de Bordeaux, ralf.klasing@labri.fr
[3]Inria – LIAFA – Paris Diderot University, adrian.kosowski@inria.fr
[4]LIF – Aix Marseille University, przemyslaw.uznanski@lif.univ-mrs.fr

Abstract

We consider the problem of deterministic load balancing of tokens in the discrete model. A set of n processors is connected into a d-regular undirected network. In every time step, each processor exchanges some of its tokens with each of its neighbours in the network. The goal is to minimize the discrepancy between the number of tokens on the most-loaded and the least-loaded processor as quickly as possible.

Rabani *et al.* (1998) present a general technique for the analysis of a wide class of discrete load balancing algorithms. Their approach is to characterize the deviation between the actual loads of a discrete balancing algorithm with the distribution generated by a related Markov chain. The Markov chain can also be regarded as the underlying model of a continuous diffusion algorithm. Rabani *et al.* showed that after time $T = \mathcal{O}(\log(Kn)/\mu)$ any algorithm of their class achieves a discrepancy of $\mathcal{O}(d \log n/\mu)$, where μ is the spectral gap of the transition matrix of the graph, and K is the initial load discrepancy in the system.

In this work we identify some natural additional conditions on deterministic balancing algorithms, resulting in a class of algorithms reaching a smaller discrepancy. Specifically, we introduce the notion of *cumulatively fair* load-balancing algorithms where the total number of tokens sent over any outgoing edge of a node is the same for every interval of consecutive time steps. We prove that algorithms, which are cumulatively fair and where every node retains a sufficient part its load in each step, achieve a discrepancy of $\mathcal{O}(\min\{d\sqrt{\log n/\mu}, d\sqrt{n}\})$ in time $\mathcal{O}(T)$. We also show that in general neither of these assumptions may be omitted without increasing discrepancy. We then show that any cumulatively fair scheme satisfying some additional assumptions achieves a discrepancy of $\mathcal{O}(d)$ almost as quickly as the continuous diffusion process.

1 Introduction

In this paper, we analyze *diffusion-based load balancing algorithms*. We assume that the processors are connected by an arbitrary d-regular graph G. At the beginning, every node has a certain number of tokens, representing its initial load. In general, diffusion-based load balancing algorithms operate in parallel, in synchronous steps. In each step, every node balances its load with *all* of its neighbours. The goal is to distribute the tokens in the network graph as evenly as possible. More precisely, we aim at minimizing the *discrepancy* (also known as *smoothness*), which is defined as the difference between the maximum load and the minimum load, taken over all nodes of the network.

One distinguishes between *continuous* load balancing models, in which load can be split arbitrarily, and the much more realistic *discrete* model, in which load is modeled by tokens which cannot be split. In the former case, the standard continuous diffusion algorithm works as follows. Every node u having load $x(u)$ considers all of its neighbours at the same time, and sends $x(u)/(d+1)$ load to each of its d neighbours, keeping $x(u)/(d+1)$ load for itself. This process may also be implemented more efficiently: For each neighbour v of u, node u calculates the load difference $x(u) - x(v)$ between them and divides the outcome by $d+1$. If $x(u) - x(v) > 0$, then it sends exactly $(x(u) - x(v))/(d+1)$ load to v. It is well-known (*cf. e.g.* [18]) that the load in the continuous model will eventually be perfectly balanced. In the discrete case with indivisible tokens, exact simulation of the continuous process is not possible. A node u may instead try to round the value $(x(u) - x(v))/(d+1)$ up or down to an integer. Discrete balancing approaches are, in general, much harder to analyze than continuous algorithms.

In [16], Rabani *et al.* suggest a framework to analyze a wide class of discrete neighbourhood load balancing algorithms in regular graphs (it can be adapted to non-regular graphs). The scheme compares the discrete balancing algorithm with its the continuous version, and the difference is used to bound the so-called error that occurs due to the rounding. Their results holds for *round-fair* algorithms where any node u sends either $\lfloor x_t(u)/d \rfloor$ or $\lceil x_t(u)/d \rceil$ tokens over its edges. They show that the discrepancy is bounded by $\mathcal{O}(d \log n/\mu)$ after $T = \mathcal{O}(\log(Kn)/\mu)$ steps, where μ is the eigenvalue gap of the transition matrix of the underlying Markov Chain. The scheme applies to any discrete load balancing scheme which, at every time step, rounds the load which would be exchanged in the continuous diffusion process by a given pair of nodes to one of the nearest integers, either up or down.

The time T in the above bound is also the time in which a continuous algorithm balances the system load (more or less) completely. T can also be regarded as the *mixing time* of a random walk on G, which is the time it takes for a random walk to be on every node with almost the same probability. Within the class of schemes considered in [16], the bound of $\mathcal{O}(d \log n/\mu)$ on discrepancy cannot be improved for many important graph classes, such as constant-degree expanders. Since the work [16], many different refinements and variants of this approach have been proposed [4, 9, 10, 17], as well as extensions to other models, including systems with non-uniform tokens [4] and non-uniform machines [2].

Our contribution. The main contribution of our paper is an continuation of the work of [16]. We suggest two general frameworks that can be used to analyze large classes of diffusion algorithms. Within the classes we consider, we provide the first systematic improvement with respect to the rounding strategy used in [16], achieving deterministic discrete load-balancing schemes with smaller discrepancy and in the same time $\mathcal{O}(T)$.

The first class of algorithms that we consider is that of is *cumulatively fair* algorithms. An algorithm is cumulatively fair if, for every interval of consecutive time steps, the total number of tokens it sends out over the edges of one node differs by at most a small constant. The class of cumulatively fair algorithms contains many stateless deterministic algorithms. It can be regarded as a restriction of the round-fairness of [16] and a generalization of the rounding used by algorithms distributing the tokens in a round-robin[1]

[1]Every node maintains a list of outgoing edges and the i-th token, distributed by a node, is sent over the outgoing edge having the position $i \bmod$ (# of edges in the list).

fashion. We show (Theorem 3.2) that algorithms satisfying the cumulative fairness condition achieve $\mathcal{O}(d \cdot \min\{\sqrt{\log n/\mu}, \sqrt{n}\})$-discrepancy in $\mathcal{O}(T)$ time steps, if at least half of the edges in the graph are self-loop edges. This provides an improvement with respect to the general result from [16]: for example, for expanders, the achieved discrepancy after time $\mathcal{O}(T)$ is $\mathcal{O}(\sqrt{\log n})$, as opposed to $\Theta(\log n)$.

The second class of algorithms we consider is a subset of cumulative fair algorithms, we call them *strictly fair and self-preferring*. A cumulatively fair algorithm is called *strictly fair* if, in every step, the number of tokens that are sent out over every edge incident to a node differs by at most one. It is called *self-preferring* if for each node, at least one self-loop edge adjacent to it receives at least as many tokevn cns as any adjacent non-self-loop edge. For strictly fair and self-preferring algorithms, we show that an $\mathcal{O}(d)$-discrepancy is achieved within $\mathcal{O}(T + d \log^2 n/\mu)$ time steps. Note that the only other result in the literature known to the authors which achieves a discrepancy of $\mathcal{O}(d)$ in $\mathcal{O}(T)$ steps in the diffusive model is the one of [4]; however, their algorithm relies on extensive communication, computation and memory capabilities of the nodes, as well as certain additional assumptions on minimum load of nodes. (*cf.* Table 1 for a comparison of properties). In addition to these upper bounds, we show several lower bounds illustrating the necessity of the fairness and self-preference conditions of the classes defined in this paper.

There are many important load balancing algorithms satisfying our first condition of cumulative fairness. To give some examples, SEND$(f(x))$, is the class of stateless algorithms in which the number of tokens sent out by a node to each of its neighbours is given by the same deterministic function $f : \mathbf{N} \to \mathbf{N}$ of its current load x. The so-called ROTOR-ROUTER model (also referred to in the literature as the Propp model, [3,6,11]) which uses a simple round-robin approach to distribute the tokens to neighbours also falls into this class. Algorithms which meet our second conditions (strict fairness and self-preference) include some variants of the rotor-router approach (*e.g.*, one variant which we denote by ROTOR-ROUTER$*$), as well as some algorithms from the stateless SEND family. One particularly representative example of such an SEND-type algorithm is the algorithm in which every node divides its load by $2d$, rounds this value to the nearest integer, and send it to every neighbour.

Our techniques for the analysis of cumulatively fair algorithms rely on a comparison between our discrete process and of the continuous process. The later can also be regarded as a Markovian process (random walk) which is governed by the transition matrix of the graph. We calculate the total deviation (of any cumulatively fair algorithm) to the continuous process as done in [16]. However, instead of doing it step-by-step as in [16], the comparison is done over long time intervals. In particular, we present the first analysis connecting this deviation to the mixing time of the Markov process. The analysis of the cumulative deviation was used in [7, 13, 20] for the special case of the ROTOR-ROUTER but in contexts different from load balancing. The authors of [13] present a follow-up application of the approach presented here in which they bound the cover time of the ROTOR-ROUTER for specific graphs. For the analysis of strictly fair and self-preferring algorithms we combine the algebraic techniques used for the analysis of cumulatively fair algorithms with a potential function approach.

1.1 Related Work

Herein we only consider related results for load balancing in the discrete diffusive and balancing circuit models, and some results for the rotor-router model which are relevant to our work.

Diffusive load balancing. Discrete load balancing processes have been studied in numerous works since [16]. The authors of [9] propose a deterministic load balancing process in which the continuous load transfer on each edge is rounded up or down deterministically, such that the sum of the rounding errors on each edge up to an arbitrary step t is bounded by a constant. This property is called the *bounded-error property*. Then they show that after T steps any process with bounded-error property achieves a discrepancy of $\mathcal{O}(\log^{3/2} n)$ for hypercubes and $\mathcal{O}(1)$ for constant-degree tori. There are no similar results for other graph classes. Note that the algorithm of [9] has the problem that the outgoing demand of a node might exceed its available load,

Algorithm	Discrepancy after time $\mathcal{O}(T)$	Time to reach $\mathcal{O}(d)$-balance	Ref.	D	SL	NL	NC
Discrete diffusion with arbitrary rounding on edges	$\mathcal{O}\left(\frac{d\log n}{\mu}\right)$		[16]	✓	✓	✓	✓
Randomized distribution of extra tokens by vertices	$\mathcal{O}\left(d\sqrt{\log n}+\sqrt{\frac{d\log n\log d}{\mu}}\right)$		[5]	X	✓	✓	✓
	$\mathcal{O}\left(d^2\sqrt{\log n}\right)$		[17]				
Randomized rounding to nearest integers on edges	$\mathcal{O}\left(\sqrt{d\log n}\right)$		[17]	X	✓	X	✓
Cumulatively fair algorithms	$\mathcal{O}\left(d\min\left\{\sqrt{\frac{\log n}{\mu}},\sqrt{n}\right\}\right)$		Thm 3.2				
• ROTOR-ROUTER	"		Thm 3.2	✓	✓	✓	✓
• SEND$\left(\left\lfloor\frac{x_t(u)}{cd}\right\rfloor\right)$, for any $c\geq 2$:	"		Thm 3.2	✓	✓	✓	✓
• **Strictly fair and self-preferring algorithms**	"	$O\left(T+\frac{d\log^2 n}{\mu}\right)$	Thm 3.4				
• ROTOR-ROUTER$*$	"	"	Thm 3.4	✓	✓	✓	✓
• SEND$\left(\left\lceil\frac{x_t(u)}{2d}-\frac{1}{2}\right\rceil\right)$	"	"	Thm 3.4	✓	✓	✓	✓
• SEND$\left(\left\lceil\frac{x_t(u)}{3d}-\frac{2}{3}\right\rceil\right)$	"	$\mathcal{O}\left(T+\frac{\log^2 n}{\mu}\right)$	Thm 3.4	✓	✓	✓	✓
• Computation based on continuous diffusion	$\Theta(d)$	$\mathcal{O}(T)$	[4]	✓	X	X	X

Legend: D — Deterministic process; SL — Stateless process; NL — Cannot produce negative loads;
NC — No additional communication required.
n — number of nodes of the graph, T — load balancing time of the continuous diffusion process,
μ — spectral gap of graph transition matrix, diam — graph diameter.

Table 1: A comparison of the discrepancy of load-balancing algorithms in the diffusive model for d-regular graphs. In all result statements, we assume that the graph is augmented with at least d self-loops per node.

leading to so-called *negative load*.

In [3], the authors consider ROTOR-ROUTER-type walks as a model for load balancing. It is assumed that half of the edges of every node are self-loops. The authors present an algorithm which falls in the class of *bounded-error diffusion processes* introduced in [9]. This results in discrepancy bounds of $\mathcal{O}(\log^{3/2} n)$ and $\mathcal{O}(1)$ for hypercube and r-dimensional torus with $r=\mathcal{O}(1)$. In [2], the authors consider the diffusion algorithms that always round down for heterogenous networks. They also show that a better load balance can be obtained when the algorithm is allowed to run longer than T steps.

In [4], the authors propose an algorithm that achieves discrepancy of $2d$ after T steps for any graph. For every edge e and step t, their algorithm calculates the number of tokens that should be sent over e in t such that the total number of tokens forwarded over e (over the first t steps) stays as close as possible to the amount of load that is sent by the continuous algorithm over e during the first t steps. However, their algorithm can result in negative load when the initial load of any node is not sufficiently large and it has to calculate the number of tokens that the continuous algorithm sends over all edges. Note that the algorithm presented in this paper has to simulate the continuous algorithm in order to calculate the load that it has to transfer over any edge, whereas our algorithms are much easier and they do not need any additional information, not even the load of their neighbours.

There are several publications that suggest randomized rounding schemes [1, 3–5, 9, 17] to convert the continuous load that is transferred over an edge into discrete load. The algorithm of [5] calculates the number of *additional* tokens (the difference between the continuous flow forwarded over edges and the number of tokens forwarded by the discrete algorithm after rounding *down*). All additional tokens are sent to randomly chosen neighbours. Their discrepancy bounds after T steps are $\mathcal{O}(d\log\log n/\mu)$ and $\mathcal{O}(d\sqrt{\log n}+\sqrt{d\log n\log d/\mu})$ for d-regular graphs, $\mathcal{O}(d\log\log n)$ for expanders, $\mathcal{O}(\log n)$ for hypercubes, and $\mathcal{O}(\sqrt{\log n})$ for tori. The authors of [17] present two randomized algorithms for the diffusive model. They achieve after $\mathcal{O}(T)$ time a discrepancy of $\mathcal{O}(d^2\sqrt{\log n})$ by first sending $\lfloor x(u)/(d+1)\rfloor$ tokens to every neighbour and itself, and afterwards by distributing the remaining tokens randomly. Additionally, they provide an algorithm which achieves after $\mathcal{O}(T)$ time a discrepancy of $\mathcal{O}(\sqrt{d\log n})$ by rounding the flow sent over edges randomly to the nearest integers which might cause negative loads (see Section 1.1 for

a detailed discussion). For a comparison with our results see Table 1.

Dimension exchange model. In the Dimension Exchange model, the nodes are only allowed to balance with one neighbour at a time. Whereas for all diffusion algorithms considered so far the discrepancy in the diffusion model is at least d, dimension exchange algorithms are able to balance the load up to an additive constant. In [10] the authors consider a discrete dimension exchange algorithm for the matching model. Every node i that is connected to a matching edge calculates the load difference over that edge. If that value is positive, the algorithm rounds it up or down, each with a probability one half. This result is improved in [17], where the authors show that a constant final discrepancy can be achieved within $\mathcal{O}(T)$ steps for regular graphs in the random matching model, and constant-degree regular graphs in the periodic matching (balancing circuit) model.

Rotor-router walks. Originally introduced in [15], the *rotor-router walk* model was employed by Jim Propp for derandomizing the random walk, thereby frequently appearing under the alternative names of *Propp machines* and *deterministic random walks* [6, 8, 11, 12]. In the rotor-router model, the nodes send their tokens out in a round-robin fashion. It is assumed that the edges of the nodes are cyclically ordered, and that every node is equipped with a rotor which points to one of its edges. Every node first sends one token over the edge pointed to by the rotor. The rotor is moved to the next edge which will be used by the next token, and so on, until all tokens of the node have been sent out over one of the edges. 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. The rotor-router model can be used for load balancing, and directly fits into the framework we consider in this paper. In [3], the authors study a lazy version of the rotor-router process (half of the edges are self-loops) for load balancing. They prove that the rotor walk falls in the class of *bounded-error diffusion processes* introduced in [9]. Using this fact they obtain discrepancy bounds of $\mathcal{O}(\log^{3/2} n)$ and $\mathcal{O}(1)$ for the hypercube and r-dimensional torus with $r = \mathcal{O}(1)$, respectively, which improve the best existing bounds of $\mathcal{O}(\log^2 n)$ and $\mathcal{O}(n^{1/r})$.

2 Model

The input of the load-balancing process is a symmetric and directed regular graph $G = (V, E)$ with n nodes. Every node has out-degree and in-degree d. We have $m \in \mathbf{N}$ indivisible tokens (workload) which are arbitrarily distributed over the nodes of the network. For simplicity of notation only, we assume that initially G does not contain multiple edges. In general, the nodes of the graph may be treated as anonymous, and no node identifiers will be required. In some algorithms, we will assume that each node u maintains a local ordering of its neighbours within its neighbourhood, which is persistent over time. For the sake of analysis only, we number the nodes with consecutive integers from 1 to n, and identify nodes with their labels.

The time is divided into synchronized steps. Let $\mathbf{x}_t = (x_t(1), \ldots, x_t(n))$ be the *load vector* at the beginning of step t, where $x_t(u)$ corresponds to the load of node u at the beginning of step t. In particular, $\mathbf{x}_1$ denotes the initial load distribution and $\bar{\mathbf{x}}$ is the real-valued vector resulting when every node has achieved average load, $\bar{\mathbf{x}}(u) = \frac{1}{n}\sum_{u\in V} x_1(u) \equiv \bar{x}$, for all $u \in V$. Note that the total load summed over all nodes does not change over time. We will denote by K the maximal initial discrepancy in $\mathbf{x}_1$, *i.e.*, $K = \max_{u\in V} x_1(u) - \min_{u\in V} x_1(u)$.

Methods of retaining load at a node. We introduce two different ways which enable nodes to keep a fraction of their own load, instead of distributing it to their neighbours: *self-loops* and so-called *remainders*. The remainders are necessary to show in Proposition 6.6 that strictly fair and self-preferring algorithms are also cumulative fair.

Formally, in order to introduce self-loops, we transform G into the graph $G^+ = (V, E \cup E^\circ)$ by adding d° self-loops to every node. For a fixed node $u \in V$, let $E^\circ_u = \{e_1(u,u), \ldots, e_{d^\circ}(u,u)\}$ denote the set

of self-loops of u and let E_u denote the outgoing edges of u in G. We assume $d^\circ = O(d)$. We can now define $E_u^+ = E_u^\circ \cup E_u$ and $E^\circ = \bigcup_{u \in V} E_u^\circ$. In the following, we call G the *original graph* and G^+ the *balancing graph*. We remark again that the balancing graph is introduced for purposes of analysis, only, and is completely transparent from the perspective of algorithm design. We also define $d^+ = d + d^\circ$ as the degree of any node in G^+. $N(u)$ is the set of direct neighbours of u in G^+, *i.e.*, it contains all neighbours of u in G including u itself (because of the self-loops).

For a fixed edge $e = (u, v) \in E^+$ let $f_t(e)$ be the number of tokens which u sends to v in step t. In particular, let $f_t(u, u) = \sum_{e \in E_u^\circ} f_t(e)$. Let $F_t(e)$ denote the cumulative load sent from u to v in steps $1, \ldots, t$, i.e, $F_t(e) = \sum_{\tau \leq t} f_\tau(e)$. For a fixed node u, we define $f_t^{out}(u) = \sum_{v \in N(u)} f_t(u, v)$ to be the number of tokens (or flow) leaving u in step t. The incoming flow is then defined as $f_t^{in}(u) = \sum_{\{v : u \in N(v)\}} f_t(v, u)$. We define the cumulative incoming and outgoing flows $F_t^{out}(u)$ and $F_t^{in}(u)$ accordingly, and $\mathbf{F}_t^{out}$ and $\mathbf{F}_t^{in}$ are defined as the vectors of the (cumulative) outgoing and incoming flow. Let the remainder $r_t(u)$ of node u in step t is the number of tokens of u that will not participate in the load distribution over its outgoing edges and self-loops. Note that for all $u \in V$ and all steps t

$$x_1(u) + F_{t-1}^{in}(u) = r_t(u) + F_t^{out}(u). \tag{1}$$

Moreover,

$$x_t(u) = f_t^{out}(u) + r_t(u). \tag{2}$$

Then, $\mathbf{r}_t = (r_t(1), \ldots, r_t(n))$ denotes the *remainder vector* at step t, where $r_t(i)$ is the number of tokens kept by the i'th node of G in step t. We will denote by r the upper bound on the maximum remainder of an algorithm, satisfying $|r_t(u)| \leq r \leq d^+$ for every time step t and every $u \in V$.

Finally, the *discrepancy* is defined as the load difference between the node with the *highest load* and the node with the *lowest load*. The *balancedness* of an algorithm is defined as the gap between the node with the highest load and the *average load*. An algorithm A is called *stateless* if the load it sends over edges in any step depends solely on the load of the node at this time step.

3 Main Results

We are now ready to formally state the main theorems of the paper. Recall that $T = \mathcal{O}(\log K + \log n/\mu)$ is the balancing time of the continuous diffusion algorithm if the initial discrepancy is K, and d is the number of outgoing edges of each node.

3.1 Cummulative Fair Algorithms

Our first result concerns cumulatively fair algorithms. For these algorithms the *cumulative* flow that is sent out over every edge of u (including the self-loops) up to step t can differ by at most δ. Additionally, the number of tokens not being distributed over self-loops or outgoing edges has to be smaller than the degree of the balancing graph.

Definition 3.1. *An algorithm is called* cumulatively δ-fair *if for all $t \in \mathbf{N}$, $u \in V$ we have*

- *all edges (including self-loops) $e_1, e_2 \in E_u^+$ satisfy $|F_t(e_1) - F_t(e_2)| \leq \delta$, and*
- *the remainder kept on all nodes is bounded by $|r_t(u)| \leq r \leq d^+$.*

Note that *round-fair* algorithms (as defined in [16]) are not necessarily commutatively δ-fair for any constant δ.

Theorem 3.2. *Let δ be a constant and let A be any cumulatively δ-fair algorithm. For any d-regular input graph G, after $\mathcal{O}(\log(Kn)/\mu)$ time steps A achieves:*

(i) A discrepancy of $\mathcal{O}\left((\delta+1)d \cdot \sqrt{\log n/\mu}\right)$ for $d^+ \geq 2d$

(ii) A discrepancy of $\mathcal{O}\left((\delta+1)d \cdot \sqrt{n}\right)$ for $d^+ \geq 2d$

(iii) A discrepancy of $\mathcal{O}\left((\delta+1)d \cdot \log n/\mu\right)$ for arbitrary $d^+ \geq d+1$,

where d^+ is the degree of the balancing graph (including self-loops) used by A.

For constant δ and at least d self-loops, the results of Theorem 3.2 (i) show a better discrepancy after T steps compared to the result of [16]. For a smaller number of self-loops, we were unable to show the same discrepancy without increasing the required time. Theorem 3.2 (ii) provides an improvement for graphs with a bad expansion (small eigenvalue gap), such as cycles. The proof of the theorem is deferred to Section 5.

We remark that we can neither drop the condition of cumulative fairness (Theorem 7.1), nor remove self-loops completely (7.3), in the above theorem. In Theorem 7.1 we show that there are round-fair processes (satisfying the constraints of [16]), yet have $\Omega(\operatorname{diam} d)$ discrepancy. Such discrepancy is worse than the one given by Theorem 3.2 for many graph classes. In Theorem 7.3 we show for a specific cumulatively 1-fair algorithm not using any self-loops (ROTOR-ROUTER), that it cannot balance better than $\Omega(dn)$ on a cycle with n nodes.

3.2 Strictly Fair and Self-preferring Algorithms

The notion of *strictly δ-fairness* restricts a δ-cumulatively fair algorithm to distribute the load fairly among all edges such that the load difference of every pair of edges is at most 1 in every step. We remark that in the definition of strictly fair algorithms, the cumulative fairness-type condition is only imposed on outgoing edges, and does not include self-loops. A strictly fair algorithm is additionally *self-preferring* if it favours self-loop edges over original edges. As we will see in Theorem 3.4, the "more self-preferring" an algorithm is, the faster it balances. For the ease of presentation only, we assume δ to be a constant.

Definition 3.3. *An algorithm is called* strictly δ-fair *for some $\delta = O(1)$ if for all $t \in \mathbf{N}$, $u \in V$ we have*

- *all outgoing edges $e_1, e_2 \in E_u$ satisfy $|F_t(e_1) - F_t(e_2)| \leq \delta$,*
- *all edges (including self-loops) receive $\lfloor x_t(u)/d^+ \rfloor$ or $\lceil x_t(u)/d^+ \rceil$ many tokens in step t, and*
- *no remainder is kept on nodes ($r_t(u) = r = 0$).*

*If additionally for $s \leq d^\circ$ and every step t at least s self-loops receive $\lceil x_t(u)/d^+ \rceil$ many tokens, then a strictly δ-*fair *algorithm is also called s-*self-preferring.

For examples we refer to Section 4. In Proposition 6.6 we observe that strictly δ-fair algorithms are a subclass of cumulatively δ-fair algorithms with a remainder larger that zero. Without the definition of the remainder in Theorem 3.1 strictly δ-fair algorithms are not a subclass of cumulatively δ-fair algorithms.

The next theorem shows that strictly fair and self-preferring algorithms achieve a smaller discrepancy of $O(d)$, giving the algorithm a slightly longer time for balancing. The proof of the theorem is provided in Section 6. Note that due to Proposition 6.6, Theorem 3.2 also applies to the strictly fair and self-preferring algorithms.

Theorem 3.4. *Let $1 \leq s \leq (d^+ - d)$ and let A be a strictly δ-fair algorithm which is s-self-preferring. Then A achieves a discrepancy of $((2\delta+1)d^+ + 4d^\circ)$ after time $O\left(\log K + ds^{-1} \cdot \log^2 n/\mu\right)$.*

We note that large values of s ($s = \Omega(d)$) increase the speed of the balancing process. However, as s is upper-bounded by the number of self-loops at a node, we need to impose $\Omega(d)$ self-loops to take advantage of this property. Unfortunately, we are not able to argue that strictly fairness and self-preference are necessary conditions to show the results of Theorem 3.4. In Lemma 7.2 (Section 7) we provide a lower bound of $\Omega(d)$ on the discrepancy of any stateless algorithms, the bound is independent of the balancing time. Since the class of strictly fair and self-preferring algorithms contains many stateless algorithms, this also means that the bound on the discrepancy Theorem 3.4 cannot be improved without further restrictions on the class.

4 Specific Algorithms and Results

Here we consider two specific families of algorithms. One is the class of stateless algorithms called SEND, while the other consists of variants of the rotor-router model.

Stateless algorithms. We define the class of algorithms SEND$(g(x_t(u)))$ to be the class of algorithms for which every node $u \in V$ sends $g(x_t(u)) \in \mathbf{N}$ many tokens over every outgoing edge $e = (u, v) \in E_u$ with $v \in V$. All algorithms presented in this section are round-fair, *i.e.* they fulfill $g(x_t(u)) \in \{\lfloor x_t(u)/d^+ \rfloor, \lceil x_t(u)/d^+ \rceil\}$. Specific examples of SEND-class algorithms, which are also mentioned in Table 1, are given below:

1. SEND$(\lfloor x_t(u)/d^+ \rfloor)$, for fixed $d^+ \geq 2d$. This algorithm sends the same number of tokens over every edge (and self-loop) in every step, the remaining tokens are kept in the remainder vector. We observe that this algorithm is cumulatively 0-fair but not strictly fair and Theorem 3.2 applies.

2. SEND$(\lceil x_t(u)/2d - 1/2 \rceil)$. This algorithm rounds $x_t(u)/(2d)$ to the nearest integer. This algorithm can be modelled by using d self-loops in the following way. First node u sends $\lfloor x_t(u)/d^+ \rfloor$ tokens to each neighbour. If $x_t(u) \bmod d^+ > d^+/2$, u sends one extra token to every neighbour. This algorithm is 0-strictly fair and 1-self-preferring. Following Theorem 2.6, the algorithm achieves a discrepancy of $(10d)$ in $O\left(\log K + d \log^2 n/\mu\right)$ time steps.

3. SEND$(\lceil x_t(u)/3d - 2/3 \rceil)$ This algorithm can be modelled similarly to the previous algorithm using use exactly $2d$ self-loops.

   This algorithm approaches a discrepancy close to $\bar{x}$ faster since it has a self-preference of $\Omega(d)$, but its discrepancy is slightly worse. We observe that this algorithm is 0-strictly fair and $\lceil d/3 \rceil$-self-preferring. Following Theorem 3.4, the algorithm achieves a discrepancy of $(17d)$ in $O\left(\log K + \log^2 n/\mu\right)$ time steps.

Rotor-type algorithms.

1. ROTOR-ROUTER. This algorithm follows the standard round-robin process on the balancing graph with at least d self-loops. The edges of the nodes can be considered in an arbitrary order. It is easy to see that ROTOR-ROUTER belongs to the class of 1-cumulatively fair algorithms and Theorem 3.2 applies.

2. ROTOR-ROUTER$*$ This algorithm maintains exactly $d - 1$ self-loops, together with one one special self-loop. The special self-loop always receives $\lceil x_t(u)(/2d) \rceil$ tokens. The remaining tokens are distributed fairly using a rotor-router on the outgoing edges and the $d - 1$ self-loops. This algorithm is 1-strictly fair and (due to the existence of the special self-loop) 1-self-preferring. By Theorem 3.4, it achieves a discrepancy of $(14d)$ in $O\left(\log K + d \log^2 n/\mu\right)$ time steps. The bounds (i) and (ii) of Theorem 2.5 apply for this algorithm.

3. ROTOR-ROUTER$*$SINGLELOOP This algorithm is similar to ROTOR-ROUTER$*$ without self-loops but the one special self-loop. The special loop always receives $\lceil x_t(u)/d + 1 \rceil$ tokens. It is 1-strictly fair and 1-self-preferring. By Theorem 3.4, it achieves a discrepancy of $(5d+9)$ in $O\left(\log K + d \log^2 n/\mu\right)$ time steps. The bounds (i) and (ii) of Theorem 2.5 do not apply for this algorithm.

5 Cumulatively Fair Algorithms

In this section, we analyze the broad class of cumulatively fair load balancing algorithms, formally defined in Section 2, with the goal of proving Theorem 3.2. The core idea of the proof is to regard the balancing process over several steps and to observe that the cumulative load of the nodes is closely related to a random walk of all tokens. The difference to the random walk can be bounded by a small corrective vector depending on δ and r. Before continuing with the proof, we require some further definitions.

5.1 Preliminaries

Let $\mathbf{P}$ denote the transition matrix of a random walk of G^+. Let $\mathbf{P}(u,v)$ denote the one-step probability for the walk to go from u to v. Then $\mathbf{P}(u,v) = 1/d^+$ if $(u,v) \in E$, $\mathbf{P}(u,v) = d^\circ/d^+$ if $u = v$, and $\mathbf{P}(u,v) = 0$ otherwise.

Let μ be the eigenvalue gap of P. We define $\mathbf{P}^t$ to be the t-steps transition matrix, *i.e.*, $\mathbf{P}^t = \mathbf{P} \cdot \mathbf{P}^{t-1}$. We define the *steady-state* distribution as $\mathbf{P}^\infty = \lim_{t\to\infty} \mathbf{P}^t$. Note that $\forall u, v \in V$, $\mathbf{P}^\infty(u,v) = d^+/(2|E^+|) = 1/n$. Observe that $P^\infty \cdot \mathbf{x}_1 = (\bar{x}, \bar{x}, \ldots, \bar{x})$. We can express $\mathbf{P}^t$ as $\mathbf{P}^t = \mathbf{P}^\infty + \mathbf{\Lambda}_t$, where $\mathbf{\Lambda}_t$ is the error-term calculating the difference between $\mathbf{P}^t$ and the steady-state distribution. We define similar to [14]

$$\text{diff}(t) = \max_{\|\mathbf{q}\|_1 = 1} \{\left\|(\mathbf{P}^t - \mathbf{P}^\infty)\mathbf{q}\right\|_1\} = \max_{\|\mathbf{q}\|_1 = 1} \{\|\mathbf{\Lambda}_t \mathbf{q}\|_1\}.$$

The mixing time t_{mix} is defined as $\arg\min_t\{\text{diff}(t) \leq 1/2\}$. Let $p \geq 1$. The p-norm of a vector $\mathbf{r}$ is defined as $\|\mathbf{r}\|_p = (\sum_{i=1}^n |r_i|^p)^{\frac{1}{p}}$. In particular, $\|\mathbf{r}\|_\infty$ is defined to be $\max\{r_1, \ldots, r_n\}$.

We are ready to proof the main theorem of this section. The core idea of the proof is to calculate the total deviation between any cumulatively fair algorithm and a continuous process, similar to [16]. However, instead of comparing the two processes step-by-step as in [16], the comparison is done over long time intervals.

5.2 Proof of Theorem 3.2

Proof. Fix a node $u \in V$. Note that by the definition of cumulative δ-fairness we have for all $(u,v) \in E_u^+$

$$\left| F_t(u,v) - \frac{F_t^{out}(u)}{d^+} \right| \leq \delta. \tag{3}$$

We will define a corrective vector which at time t measures the difference between the load the nodes sent over outgoing edges at time t and the load the nodes should have sent over these edges in order to ensure that every outgoing edge received the exact same (continuous) load until time t. Formally, we define the n-dimensional corrective vector $\boldsymbol{\delta}_{t,u}$ with $\delta_{t,u}(v) = F_t(v,u) - F_t^{out}(v)/d^+$ for $v \neq u$ and $\delta_{t,u}(u) = F_t(u,u) - d^\circ/d^+ \cdot F_t^{out}(u)$. The entries of the corrective vector satisfy $|\delta_{t,u}(v)| \leq \delta$ for $v \in N(u) \setminus \{u\}$, $|\delta_{t,u}(u)| \leq d^\circ\delta$, and $\delta_{t,u}(v) = 0$ for $v \notin N(u)$. Consequently, $\|\boldsymbol{\delta}_{t,u}\|_1 \leq \delta d^+$. Then we derive from (3) the

following bound on the incoming cumulative load of node u

$$\begin{aligned} F_t^{in}(u) &\underset{def.}{=} \sum_{v\in N(u)} F_t(v,u) = \sum_{v\in N(u)\setminus\{u\}} F_t(v,u) + F_t(u,u) \\ &\underset{(3)}{=} \sum_{v\in N(u)\setminus\{u\}} \left(\frac{1}{d^+}F_t^{out}(v) + \delta_{t,u}(v)\right) + \frac{d^\circ}{d^+}F_t^{out}(u) + \delta_{t,u}(u) \\ &= \sum_{v\in N(u)\setminus\{u\}} \frac{1}{d^+}F_t^{out}(v) + \frac{d^\circ}{d^+}F_t^{out}(u) + \left\|\boldsymbol{\delta}_{t,u}\right\|_1 . \end{aligned} \tag{4}$$

Rewriting (1) by introducing (4) we get:

$$F_t^{out}(u) = x_1(u) + \sum_{v\in N(u)\setminus\{u\}} \frac{1}{d^+}F_{t-1}^{out}(v) + \frac{d^\circ}{d^+}F_{t-1}^{out}(u) + \underbrace{\left\|\boldsymbol{\delta}_{t,u}\right\|_1 - r_t(u)}_{\varepsilon_t(u)} \tag{5}$$

We have $\left\|\varepsilon_t(u)\right\|_\infty \le \delta d^+ + r$. Rewriting (5) in vector form, we obtain

$$\mathbf{F}_t^{out} = \mathbf{x}_1 + \mathbf{P}\cdot\mathbf{F}_{t-1}^{out} + \boldsymbol{\varepsilon}_t = \sum_{0\le\tau<t}\mathbf{P}^\tau\mathbf{x}_1 + \sum_{1\le\tau\le t}\mathbf{P}^{t-\tau}\cdot\boldsymbol{\varepsilon}_\tau.$$

For any $\widehat{T} > 0$, the number of tokens leaving node u in the interval of steps $[t+1; t+\widehat{T}]$ is $\mathbf{F}_{t+\widehat{T}}^{out}(u) - \mathbf{F}_t^{out}(u)$. Hence,

$$\mathbf{F}_{t+\widehat{T}}^{out} - \mathbf{F}_t^{out} = \sum_{t\le\tau<t+\widehat{T}}\mathbf{P}^\tau\mathbf{x}_1 + \sum_{1\le\tau\le t}(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\cdot\boldsymbol{\varepsilon}_\tau + \sum_{t<\tau\le t+\widehat{T}}\mathbf{P}^{t+\widehat{T}-\tau}\cdot\boldsymbol{\varepsilon}_\tau.$$

We set $t^* = t - 4t_{mix}$ and substitue $\mathbf{P}^\tau = \mathbf{P}^\infty + \boldsymbol{\Lambda}_\tau$. We derive

$$\begin{aligned} \mathbf{F}_{t+\widehat{T}}^{out} - \mathbf{F}_t^{out} &= \sum_{t\le\tau<t+\widehat{T}}\mathbf{P}^\tau\mathbf{x}_1 + \sum_{1\le\tau\le t^*}(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau + \sum_{t^*<\tau\le t}(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau + \sum_{t<\tau\le t+\widehat{T}}\mathbf{P}^{t+\widehat{T}-\tau}\boldsymbol{\varepsilon}_\tau \\ &= \widehat{T}\mathbf{P}^\infty\mathbf{x}_1 + \sum_{t\le\tau<t+\widehat{T}}\boldsymbol{\Lambda}_\tau\mathbf{x}_1 \;+ \sum_{1\le\tau\le t^*}(\boldsymbol{\Lambda}_{t+\widehat{T}-\tau} - \boldsymbol{\Lambda}_{t-\tau})\boldsymbol{\varepsilon}_\tau + \sum_{t^*<\tau\le t}(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\cdot\boldsymbol{\varepsilon}_\tau \\ &\quad + \sum_{t<\tau\le t+\widehat{T}}\mathbf{P}^{t+\widehat{T}-\tau}\cdot\boldsymbol{\varepsilon}_\tau. \end{aligned} \tag{6}$$

In the following we use $\bar{\mathbf{x}} = \mathbf{P}^\infty\mathbf{x}_1$.

$$\begin{aligned} \left\|\sum_{t<\tau\le t+\widehat{T}}\mathbf{x}_\tau - \widehat{T}\cdot\bar{\mathbf{x}}\right\|_\infty &\underset{(2)}{=} \left\|\left(\mathbf{F}_{t+\widehat{T}}^{out} - \mathbf{F}_t^{out} + \sum_{t<\tau\le t+\widehat{T}}\mathbf{r}_\tau\right) - \widehat{T}\mathbf{P}^\infty\mathbf{x}_1\right\|_\infty \\ &\underset{(6)}{\le} \sum_{t\le\tau<t+\widehat{T}}\left\|\boldsymbol{\Lambda}_\tau\mathbf{x}_1\right\|_\infty \;+ \sum_{1\le\tau\le t^*}\left(\left\|\boldsymbol{\Lambda}_{t+\widehat{T}-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty + \left\|\boldsymbol{\Lambda}_{t-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty\right) \\ &\quad + \sum_{t^*<\tau\le t}\left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty \;+ \sum_{t<\tau\le t+\widehat{T}}\left\|\mathbf{P}^{t+\widehat{T}-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty \;+ \sum_{t<\tau\le t+\widehat{T}}\left\|\mathbf{r}_\tau\right\|_\infty \end{aligned}$$

By (i) of Lemma 8.1 it follows for $t \geq 16 \cdot \log(nK)/\mu$ that $\forall_{\tau \geq t} \left\|\mathbf{\Lambda}_\tau \mathbf{x}_1\right\|_\infty \leq 2^{-4}$, and (ii) of Lemma 8.1 implies that $\sum_{\tau \geq 4 \cdot t_{mix}} \left\|\mathbf{\Lambda}_\tau \boldsymbol{\varepsilon}_\tau\right\|_\infty \leq 2^{-3} \cdot \max_{\tau \geq 4 \cdot t_{mix}} \{\left\|\boldsymbol{\varepsilon}_\tau\right\|_\infty\}$. This results in

$$\begin{aligned}\left\| \sum_{t<\tau\leq t+\widehat{T}} \mathbf{x}_\tau - \widehat{T} \cdot \bar{\mathbf{x}} \right\|_\infty &\leq \widehat{T} \cdot 2^{-4} \ + 2 \cdot 2^{-3} \max_{1\leq\tau\leq t^*} \left\|\boldsymbol{\varepsilon}_\tau\right\|_\infty + \sum_{t^*<\tau\leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau \right\|_\infty \\ &\quad + \widehat{T}(\delta d^+ + r) \ + \widehat{T} r \\ &\leq \frac{\widehat{T}}{4} + (\delta d^+ + r) + \sum_{t^*<\tau\leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau \right\|_\infty + \widehat{T}(\delta d^+ + 2r).\end{aligned}$$

Dividing the last equation by $\widehat{T}$ yields

$$\left\| \frac{\sum_{t<\tau\leq t+\widehat{T}} \mathbf{x}_\tau}{\widehat{T}} - \bar{\mathbf{x}} \right\|_\infty \leq \frac{1}{4} + (\delta d^+ + 2r) + \frac{(\delta d^+ + r) + \sum_{t^*<\tau\leq t} \left\| (\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau \right\|_\infty}{\widehat{T}}. \quad (7)$$

In (7) we derived a bound on the difference between the average load of a node during an interval of length $\widehat{T}$ and the average load $\bar{x}$. In the remainder we bound (7) for $\widehat{T} = 1$, which resembles the load at one time step. We will relate it to the mixing time and we will use Lemma 8.1 to show that the corrective vectors become diminishingly small.

In the following, we fix an arbitrary $t \geq 16 \log(nK)/\mu$. Let w be the i'th node of V, then we define $\mathbf{w}$ to be the vector, such that $\mathbf{w}[i] = 1$ and $\forall_{j \neq i} \mathbf{w}[j] = 0$. We define $P_t(u, w)$ to be the probability that a random walk following matrix $\mathbf{P}$, initially located at $u \in V$, is located at w after t time steps.

We observe, that by fixing $\widehat{T} = 1$, the term bounded in (7) becomes $\left\|\mathbf{x}_{t+1} - \bar{\mathbf{x}}\right\|_\infty$, This corresponds, up to a constant, to the load discrepancy (*i.e.*, the load difference between the node with the highest load the node with the lowest load), the quantity we seek to bound in this theorem.
We start by bounding the term $\left\|(\mathbf{P}^{t+1-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty$ of (7), where $t^* < \tau \leq t$, denoting $a = t - \tau$ $(0 \leq a < t - t^*)$:

$$\begin{aligned}\left\|(\mathbf{P}^{t+1-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty &= \left\|(\mathbf{P}^{a+1} - \mathbf{P}^a)\boldsymbol{\varepsilon}_{t-a}\right\|_\infty = \max_{w\in V} \left| \mathbf{w}^\top \left(\mathbf{P}^{a+1} - \mathbf{P}^a\right) \boldsymbol{\varepsilon}_{t-a} \right| \\ &= \max_{w\in V} \left| \mathbf{w}^\top \left(\mathbf{P}^{a+1} - \mathbf{P}^a\right) \left(\sum_{v\in V} \boldsymbol{\varepsilon}_{t-a}(v)\mathbf{v} \right) \right| \\ &\leq \left\|\boldsymbol{\varepsilon}_{t-a}\right\|_\infty \cdot \max_{w\in V} \sum_{v\in V} \left| \mathbf{w}^\top (\mathbf{P}^{a+1} - \mathbf{P}^a)\mathbf{v} \right| \\ &\leq (\delta d^+ + r) \cdot \max_{w\in V} \sum_{v\in V} |P_{a+1}(v, w) - P_a(v, w)| \quad (8)\end{aligned}$$

Then, since the graph is regular, we have $P_t(v, w) = P_t(w, v)$.[2] From here on we divide the analysis and prove the claimed bounds by bounding (8) separately.

(i) $O\left((\delta+1)d\sqrt{\frac{\log n}{\mu}}\right)$**-discrepancy for** $d^+ \geq 2d$ **:**
For regular graphs having $P(u, u) \geq 1/2$ for all $u \in V$, we have by [14] (for $a > 0$)

$$\forall_{w\in V} \sum_{v\in V} |P_{a+1}(w, v) - P_a(w, v)| < \frac{24}{\sqrt{a}}.$$

[2]For general graphs, one can use $P_t(v, w) = (d^+(w)/d^+(v))P_t(w, v)$

(For case of $a = 0$, we have $\forall_{w\in V} \sum_{v\in V} |P_1(w,v) - P_0(w,v)| \leq 2$.)

We obtain by applying $\sum_{a=1}^{t-t^*} 1/\sqrt{a} \leq 2\sqrt{t-t^*}$

$$\sum_{0\leq a<t-t^*} \left\|(\mathbf{P}^{a+1} - \mathbf{P}^a)\boldsymbol{\varepsilon}_{t-a}\right\|_\infty < (\delta d^+ + r)\left(2 + 48\sqrt{\underbrace{t-t^*}_{4t_{mix}}}\right) \leq 98(\delta d^+ + r)\sqrt{t_{mix}}.$$

Introducing this into into the (7) and setting $\widehat{T} = 1$ yields

$$\|\mathbf{x}_{t+1} - \bar{x}\|_\infty = O\left((\delta d^+ + r)\sqrt{t_{mix}}\right) = O\left(((\delta+1)d)\sqrt{\frac{\log n}{\mu}}\right)$$

where the last equality follows from $t_{mix} = O(\frac{\log n}{\mu})$ (see [14]) and from $r \leq d^+$. (Observe that whenever an cumulatively fair algorithm has a $\delta = 0$ the bound on the remainder r has to be of order $\Omega(d)$.)

(ii) $O\left((\delta+1)d\sqrt{n}\right)$**-discrepancy for** $d^+ \geq 2d$ **:**
Let D_{a+1} be the diagonal matrix of $(P_{a+1} - P_a)$ and let X be the corresponding base change matrix.

$$\begin{aligned}
\max_{w\in V} \sum_{v\in V} |P_{a+1}(w,v) - P_a(w,v)| &= \max_{w\in V} \sum_{v\in V} |\mathbf{v}(P_{a+1} - P_a)\mathbf{w}| \\
&= \max_{w\in V} \|(P_{a+1} - P_a)\mathbf{w}\|_1 \\
&\leq \max_{\|\mathbf{w}\|_2=1} \|(P_{a+1} - P_a)\mathbf{w}\|_1 \\
&= \max_{\|\mathbf{w}\|_2=1} \left\|X^\top D_{a+1} X\mathbf{w}\right\|_1 \\
&\leq \sqrt{n}\left\|X^\top D_{a+1} X\right\|_2 \\
&= \sqrt{n}\left\|D_{a+1}X\right\|_2. \qquad (9)
\end{aligned}$$

We have

$$D_{a+1} = \begin{pmatrix} \lambda_1^{a+1} - \lambda_1^a & 0 & \dots & 0 \\ 0 & \lambda_2^{a+1} - \lambda_2^a & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n^{a+1} - \lambda_n^a \end{pmatrix}.$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of P. We note that $\lambda_1 = 1$ and $\lambda_2, \dots, \lambda_n \in [0,1]$ since $d^\circ \geq d$. Hence by plugging (9) in (8) we derive

$$\begin{aligned}
\sum_{0\leq a<t-t^*} \left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty &\leq (\delta+1)d^+ \cdot \sum_{a<t-t^*} \max_{w\in V} \sum_{v\in V} |P_{a+1}(w,v) - P_a(w,v)| \\
&\leq (\delta+1)d^+ \cdot \sum_{a<t-t^*} \sqrt{n}\left\|D_{a+1}X\right\|_2 \\
&\leq (\delta+1)d^+ \cdot \sqrt{n} \sum_{a=0}^{t-t^*} |\lambda_2^{a+1} - \lambda_2^a| \\
&= (\delta+1)d^+ \cdot \sqrt{n} \cdot (\lambda_2^0 - \lambda_2^{t-t^*}) \\
&\leq (\delta+1)d^+ \cdot \sqrt{n}
\end{aligned}$$

Introducing this into (7) and setting $\widehat{T} = 1$ yields:

$$\|\mathbf{x}_{t+1} - \bar{x}\|_\infty \leq (\delta + 1)d^+ \cdot \sqrt{n},$$

which completes the proof.

(iii) $O\left((\delta+1)\frac{d \log n}{\mu}\right)$**-discrepancy for** $d^+ \geq d + 1$**:**
Follows from (10) by setting $\widehat{T} = 1$.

□

6 Analysis of Strictly Fair and Self-preferring Algorithms

In this section we proove Theorem 3.4. We first define the following two families of potential functions, parameterized by c:

$$\phi_t(c) = \sum_{v \in V} \max\{x_t(v) - cd^+, 0\} \text{ and } \phi'_t(c) = \sum_{v \in V} \max\{cd^+ + s - x_t(v), 0\}.$$

To show the theorem we use Equation (7) from the proof of Theorem 3.2, to derive Lemma 6.1. The lemma shows that, for every node u, there exists a time step t_u in which the load of the node has a certain distance to $\bar{x}$. We will then show that the time step t_u results in a potential drop of $\phi_t(c)$ for u if the load of u was larger than cd^+.

The following lemma gives a bound on the required length of the time interval so that there is a step t_u where u has a load which is sufficiently close to $\bar{x}$. The required time is expressed as a fraction of the mixing time ($t_{mix} = \log n/\mu$). The lemma shows a tradeoff (parameter λ) between the required time and the load difference of u to $\bar{x}$.

Lemma 6.1. *Consider any cumulatively δ-fair algorithm with remainder bounded by r, and an initialization of the load balancing process with average load $\bar{x}$ and initial discrepancy K. Let $\lambda \geq 0$, and let $t \geq 16 \cdot \log(nK)/\mu$, and let $\widehat{T} = O\left(d \log n/(\mu \cdot (\lambda + 1))\right)$. Then we have:*
For all $u \in V$ there exists a time step $t' \in [t+1; t+\widehat{T}]$ such that $x_{t'}(u) \leq \bar{x} + \delta d^+ + 2r + 1/2 + \lambda$.

Proof. We build upon the proof of Theorem 3.2 (See Section 5.2). Note that by Preposition 6.6 the Theorem also holds for strictly fair and self-preferring algorithms. We will use the notation established in the proof of Theorem 3.2.

We bound the term $\sum_{t^* < \tau \leq t} \left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty$ of (7):

$$\begin{aligned}
\sum_{t^* < \tau \leq t} \left\|(\mathbf{P}^{t+\widehat{T}-\tau} - \mathbf{P}^{t-\tau})\boldsymbol{\varepsilon}_\tau\right\|_\infty &\leq \sum_{t^* < \tau \leq t} \left(\left\|(\mathbf{P}^{t+\widehat{T}-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty + \left\|\mathbf{P}^{t-\tau}\boldsymbol{\varepsilon}_\tau\right\|_\infty\right) \\
&\leq 2 \sum_{t^* < \tau \leq t} \|\boldsymbol{\varepsilon}_\tau\|_\infty \\
&\leq 2(\delta d^+ + r)(t - t^*) = 8t_{mix}(\delta d^+ + r).
\end{aligned}$$

Injecting this into the (7) and dividing by $\widehat{T}$ gives:

$$\left\|\frac{\sum_{t < \tau \leq t+\widehat{T}} \mathbf{x}_\tau}{\widehat{T}} - \bar{x}\right\|_\infty \leq \left(\delta d^+ + 2r + \frac{1}{4}\right) + \frac{(8t_{mix} + 1)(\delta d^+ + r)}{\widehat{T}}. \tag{10}$$

Since $d^\circ = O(d^+)$, $\delta = O(1)$, and $r \leq d^+$, there is constant c such that $cd \geq \delta d^+ + r$. By [14], we have $t_{mix} \leq c' \frac{\log n}{\mu}$ for some constant c'. We set $\widehat{T} \geq 36c \cdot c' \cdot \frac{d \log n}{\mu(\lambda+1)}$.
We derive from (10)

$$\left\| \frac{\sum_{t<\tau\leq t+\widehat{T}} \mathbf{x}_\tau}{\widehat{T}} - \bar{x} \right\|_\infty \leq \delta d^+ + 2r + \frac{1}{4} + \frac{9t_{mix} \cdot cd}{36c \cdot c' \cdot \frac{d\log(n)}{\mu(\lambda+1)}} \leq \delta d^+ + 2r + \frac{1}{4} + \frac{(\lambda+1)}{4} \leq \delta d^+ + 2r + \frac{1}{4} + \lambda + \frac{1}{4}.$$

Therefore, we have that the difference between the average load of any node over $\widehat{T} = \frac{c' d \log(n)}{\mu(\lambda+1)}$ steps and the average load $\bar{x}$ is for bounded by $\delta d^+ + 2r + \frac{1}{2} + \lambda$. This means that for every node u there has to be a time step $t' \in [t+1, t+\widehat{T}]$ such that $x_{t'}(u) - \bar{x} \leq \delta d^+ + 2r + \frac{1}{2} + \lambda$. This yields the claim. □

The next lemma bounds the one-step potential drop of $\phi_t(c)$ occurring on every node which has a load of more than cd^+ at time $t-1$ and has a smaller load of at most $cd^+ + s$ at time t.

Lemma 6.2 (Monotonicity of potential)**.** *Let A be strictly fair and s-self-preferring. The potential $\phi_t(c)$ is non-increasing in time and it satisfies: $\phi_t(c) \leq \phi_{t-1}(c) - \sum_{u\in V} \Delta_t(c,u)$, where:*

$$\Delta_t(c,u) = \begin{cases} \min\{x_{t-1}(u), cd^+ + s\} - \max\{x_t(u), cd^+\} & \text{if } x_{t-1}(u) > x_t(u) \text{ and } x_{t-1}(u) > cd^+ \\ & \quad \text{and } x_t(u) < cd^+ + s \\ 0 & otherwise \end{cases}$$

Proof. Fix $c \in \mathbf{N}$. At any time t, we will divide the set L of m tokens circulating in the system into two groups: the set of *black* tokens L_t^- and the set of *red* tokens L_t^+, with $L = L_t^- \cup L_t^+$. Colors of tokens persist over time unless they are explicitly recolored. For $t = 1$, for each node u we color exactly $|L_1^-(u)| = \min\{x_1(u), cd^+\}$ tokens at u black, and the remaining tokens at u red. In every time step, we follow two rules concerning token distribution:

(1) The number of black tokens leaving a node u along any edge (including self-loops) is never more than c.

(2) At the start of each subsequent step t, we recolor some red tokens to black, so that the total number of black tokens located at a node u is exactly $|L_t^-(u)| = \min\{x_t(u), cd^+\}$.

We note that both rules of token circulation are well defined. The proof proceeds by induction. To prove the correctness of rule (1) at step t, observe that, by the definition of strictly fair algorithms, for any node u we either have $x_t(u) \leq cd^+$ and then node u sends at most c tokens along each of its edges and self-loops, or $x_t(u) > cd^+$, and then node u sends at least c tokens along each of its edges and self-loops. In the first case, rule (1) is correct regardless of how u distributes tokens of different colors; in the second case, u has exactly cd^+ black tokens, and we can require that it sends exactly c of its black tokens along each of its edges and self-loops. To prove the correctness of rule (2), we note that by the correctness of rule (1) for the preceding time step, the number of black tokens arriving at u along edges and self-loops can be upper-bounded by $\min\{x_t(u), cd^+\}$. Hence, no recoloring of tokens from black to red is ever required.

We now observe that the potential $\phi_t(c)$ is by definition the number of red tokens circulating in the system at any given moment of time. Indeed, we have:

$$\phi_t(c) = \sum_{u\in V} (x_t(u) - \min\{x_t(u), cd^+\}) = m - \sum_{u\in V} |L_t^-(u)| = m - |L_t^-| = |L_t^+|.$$

The monotonicity for the potential follows immediately from the fact that no new red tokens appear in the system. To prove the potential drop, we will show that the number of tokens being recolored from red to

black at node u in step t is at least $\Delta_t(c,u)$. Indeed, suppose that at time $t-1$ we had for some node u: $x_{t-1}(u) = cd^+ + i$, for some $i \geq 1$. Then, by definition of the self-preference of algorithm A, at least $c+1$ units of load will be sent on at least $i' = \min\{i, s\}$ self-loops of u in step t. By rule (1) of the token circulation process, each of these self-loops will contain at least one red token. Thus, the number of red tokens arriving at u at time t is at least i'. On the other hand, the number of red tokens remaining after the recoloring at u in step t is precisely $\max\{x_t(u) - cd^+, 0\}$. Thus, the number of tokens recolored from red to black at u, or equivalently the potential drop induced at u, is at least

$$\max\{i' - \max\{x_t(u) - cd^+, 0\}, 0\} = \max\{\min\{x_{t-1}(u) - cd^+, s\} - \max\{x_t(u) - cd^+, 0\}, 0\} \stackrel{\text{def}}{=} \Delta_t(c,u)$$

which yields the claimed potential drop. □

The following observation extends Lemma 6.2 to intervals $[t, t']$. It estimates the the potential drop of $\phi_t(c)$ for nodes which have a load $\geq cd^+$ at time t and a load $\leq cd^+$ during one time step of the interval. The observation follows directly from Lemma 6.2; we omit its proof.

Observation 6.3. *Let A be an s-self-preferring algorithm and let $t \leq t'$ be two fixed time steps. Denote by U the subset of nodes such that for all $u \in U$ $x_t(u) \geq cd^+ + 1$ and there exists a moment of time $t_u \in [t, t']$ such that $x_{t_u}(u) \leq cd^+$. Then $\phi_{t'}(c) \leq \phi_t(c) - \sum_{u \in U} \max\{s, x_t(u) - cd^+\}$.*

The potential defined in Lemma 6.2 bounds the number of tokens above certain thresholds. Now we use $\phi'_t(c)$ to show symmetric results measuring the number of 'gaps' below certain thresholds.

Lemma 6.4. *Let A be strictly fair and s-self-preferring. The potential $\phi'_t(c)$ is non-increasing in time and it satisfies: $\phi'_t(c) \leq \phi'_{t-1}(c) - \sum_{u \in V} \Delta'_t(c,u)$, where:*

$$\Delta'_t(c,u) = \begin{cases} \min\{x_t(u), cd^+ + s\} - \max\{x_{t-1}(u), cd^+\} & \text{if } x_{t-1}(u) < x_t(u) \text{ and } x_{t-1}(u) < cd^+ + s \\ & \quad \text{and } x_t(u) > cd^+ \\ 0 & otherwise \end{cases}$$

Before proving the lemma, we remark that the potential admits a drop at node u at time t (*i.e.*, $\Delta'_t(c,u) \geq 1$) for every node u such that $x_{t-1}(u) \leq cd^+$ and $x_t(u) \geq cd^+ + 1$, for any algorithm which is at least 1-self-preferring.

Proof. The proof is similar to Lemma 6.2. Fix $c \in \mathbf{N}$. At any time t, we will divide the set L of tokens circulating in the system into two groups: the set of *black* tokens L_t^- and the set of *red* tokens L_t^+, with $L = L_t^- \cup L_t^+$. Colors of tokens persist over time unless they are explicitly recolored. For $t = 1$, for each node u we color exactly $|L_1^-(u)| = \min\{x_1(u), cd^+ + s\}$ tokens at u black, and the remaining tokens at u red. In every time step, we follow two rules concerning token distribution:

(1) The number of black tokens leaving u along outgoing edges is at most c.

(2) At the start of each subsequent step t, we recolor some red tokens to black, so that the total number of black tokens located at a node u is exactly $|L_t^-(u)| = \min\{x_t(u), cd^+ + s\}$.

We note that both rules of token circulation are well defined. The proof proceeds by induction. To prove the correctness of rule (1) at step t, observe that, by the definition of strictly fair algorithms, for any node u we either have $x_t(u) \leq cd^+$ and then node u sends at most c black tokens along each of its edges and self-loops, or $x_t(u) > cd^+$, and then node u sends over $\max\{\min\{x_t(u) - cd^+, s\}, 0\}$ many self-loops $c+1$ black tokens and c along all other edges. In the first case, rule (1) is correct regardless of how u distributes tokens of different colors; in the second case, let $s' = \max\{\min\{x_t(u) - cd^+, s\}, 0\}$. Node u has exactly $cd^+ + s'$

black tokens, and we can require that it sends exactly $c+1$ of its black tokens along s' many arbitrary self-loops, since the algorithm is s-self-preferring, and exactly c along each of its other edges. To prove the correctness of rule (2), we note that by the correctness of rule (1) for the preceding time step, the number of black tokens arriving at u along edges and self-loops can be upper-bounded by $\min\{x_t(u), cd^+ + s\}$. Hence, no recoloring of tokens from black to red is ever required.

We now observe that the potential $\phi'_t(c)$ is by definition the number of missing black tokens such that every node has $cd^+ + s$ of them. Indeed, we have:

$$\phi'_t(c) = \sum_{u\in V}(cd^+ + s - \min\{x_t(u), cd^+ + s\}) = (cd^+ + s)\cdot n - \sum_{u\in V}|L_t^-(u)| = (cd^+ + s)\cdot n - |L_t^-|.$$

The monotonicity for the potential follows immediately from the fact that no new red tokens appear in the system. To prove the claimed potential drop, we will show that the number of tokens being recolored from red to black in time step t is at least $\Delta'_t(c,u)$. Note, that a red token, which is recolored in black, will decrease the potential by 1.
Indeed, suppose that at time $t-1$ we had for a node u: $x_{t-1}(u) = cd^+ + s - i$, for an integer $i \geq 1$. Then, by the definition of the self-preference of algorithm A, at least $i' = \min\{i, s\}$ self-loops carry at most c tokens in step t. Intuitively, each of them can 'trap' a red token. By rule (1) of the token circulation process, every neighbour of u sent at most c black tokens. Thus, the number of black tokens arriving at u at time t is at most $cd^+ + s - i'$, and the number of red tokens which u receives is at least $\max\{x_t(u) - (cd^+ + s - i'), 0\}$. Therefore, for $x_{t-1}(u) < cd^+ + s$, since at most i' self-loops 'trap' a red token we have that the number of red tokens which are repainted black at node u at time t is at least (by rule (2) of recoloring)

$$\begin{aligned}
&\min\{\max\{x_t(u) - (cd^+ + s - i'), 0\}, i'\}\\
&= \min\{\max\{\ x_t(u) - (cd^+ - \min\{cd^+ - x_{t-1}(u), 0\}), 0\}, \min\{cd^+ + s - x_{t-1}(u), s\}\}\\
&= \max\{\min\{x_t(u) - x_{t-1}(u), s, x_t(u) - cd^+, cd^+ + s - x_{t-1}(u)\}, 0\} = \Delta'_t(c,u),
\end{aligned}$$

and for $x_{t-1}(u) \geq cd^+ + s$ we have $\Delta'_t(c,u) = 0$, which yields the claimed potential drop.

□

Again, the following observation follows directly from Lemma 6.4; we omit its proof.

Observation 6.5. *Let A be an s-self-preferring algorithm and let $t \leq t'$ be two fixed time steps. Denote by U the subset of nodes such that for all $u \in U$ $x_t(u) < cd^+ + s$ and there exists a moment of time $t_u \in [t, t']$ such that $x_{t_u}(u) \geq cd^+ + s$. Then $\phi'_{t'}(c) \leq \phi_t(c) - \sum_{u\in U}\max\{s, cd^+ + s - x_t(u)\}$.* □

Below we provide the proof of Theorem 3.4. The idea of the proof is as following: we will consider the potential functions $\phi_t(c)$ for decreasing values of c and analyze the time T_c it takes to decrease the potentials $\phi_t(c)$. The time bound of Theorem 3.4 is then the sum the the times T_c for suitable chosen values of c. A symmetrical argument can be used to bound $\phi'_t(c)$.

Proof of Theorem 3.4. We consider the behaviour of the process for $t \geq T = O(\frac{\log(Kn)}{\mu})$. Due to the monotonicity described in Lemma 6.2, once the maximum load in the system is below cd^+ for some c, it will stay below this threshold. In particular, this means that the maximum load will not exceed $x_m = \bar{x} + O\left(\frac{d\log n}{\mu}\right)$ after $O\left((\delta+1)d\cdot \log n/\mu\right)$ time, as shown in Theorem 3.2((iii)).

In the first part of the proof, we will show that after a further $O(\frac{d^+}{s}\frac{\log^2 n}{\mu})$ time steps, the maximum load in the network will drop below the threshold value $c_0 d^+$, where c_0 is the smallest integer such that $c_0 d^+ \geq \bar{x} + \delta d^+ + 2d^\circ + d^+/2$.

We note that if there exists a node u with load $x_t(u) \geq c_0 d^+ + 1$ at some time moment $t \geq T$, then by Lemma 6.1 with $\lambda = d^+/2 - 1/2$ (which we can apply to strictly δ-fair algorithms, putting $r = d^\circ$, by Proposition 6.6), there exists some time step $t' \in [t+1; t+\widehat{T}]$, where $\widehat{T} = O\left(\frac{\log n}{\mu}\right)$, such that we have $x_{t'}(u) \leq c_0 d^+$. We then obtain directly from Lemma 6.3 that such a load change for node u results in the decrease of potential $\phi(c_0)$ in the time interval $[t+1; t+\widehat{T}]$. Since the potential $\phi(c_0)$ is non-increasing and non-negative, it follows that eventually the load of all nodes must be below the threshold $c_0 d^+$.

In order to prove that the load of all nodes drop below $c_0 d^+$ within $O(\frac{d^+}{s}\frac{\log^2 n}{\mu})$ time steps after time T, we apply a more involved potential-decrease argument based on the parallel drop of multiple potentials $\phi(c)$, for $c \in \{c_0, c_0+1, \ldots, c_1\}$. Here, c_1 is the smallest integer such that $c_1 d^+ \geq x_m = \bar{x} + O\left(\frac{d \log n}{\mu}\right)$.

We now partition the execution of our process into phases of duration t_p, $p = 1, 2, 3, \ldots, p_f$, such that, at the end of moment $T_p = T + t_1 + \ldots + t_p$ (end of the p'th phase), the following condition is satisfied:

$$\phi_{T_p}(c) \leq 4^{(c_1-c)} 2^{-p} (c_1 - c_0) d^+ n, \quad \text{for all } c_0 \leq c \leq c_1. \tag{11}$$

As we have remarked, the values of time t_p are well defined, since eventually the potentials $\phi(c)$ drop to 0, for all $c_0 \leq c \leq c_1$. Our goal is now to bound the ending time T_{p_f} of phase p_f, where:

$$p_f = 2(c_1 - c_0) + \lceil \log((c_1 - c_0) d^+ n) \rceil + 1. \tag{12}$$

At the end of this phase, we will have by (11) that $\phi_{T_{p_f}}(c_0) \leq 1/2$, hence $\phi_{T_{p_f}}(c_0) = 0$, and so there are no nodes having load exceeding $c_0 d^+$.

We now proceed to show the following bounds on the duration of each phase t_p:

$$t_p = O\left(\frac{d^+}{s} \frac{d \log n}{\mu \cdot \max\{(2(c_1 - c_0) - p) d^+ + 1, d^+/2 + 1\}}\right). \tag{13}$$

For any fixed p, assume that bound (13) holds for all phases before p. For phase p, the proof proceeds by induction with respect to c, in decreasing order of values: $c = c_1, c_1 - 1, \ldots, c_0$.

First, we consider values of $c \geq c_1 - p/2$. For a fixed c, following (11) we denote $b = 4^{(c_1-c)} 2^{-p} (c_1 - c_0) d^+ n$. Knowing that $\phi_{T_{p-1}}(c) \leq 2b$, $\phi_{T_{p-1}}(c+1) \leq b/2$, and by the inductive assumption $\phi_{T_p}(c+1) \leq b/4$, we will show that $\phi_{T_p}(c) \leq b$. Let $s_t(c) := \phi_t(c) - \phi_t(c+1)$; intuitively, $s_t(c)$ can be seen as total the number of tokens in the system which are "stacked" on their respective nodes at heights between $cd^+ + 1$ and $(c+1)d^+$. For $t \in [T_{p-1}; T_p]$, $s_t(c)$ satisfies the following bound:

$$s_t(c) = \phi_t(c) - \phi_t(c+1) \geq \phi_t(c) - \phi_{T_{p-1}}(c+1) \geq \phi_t(c) - \tfrac{b}{2}. \tag{14}$$

For any such time moment t consider the set of nodes with load at least cd^+ at time t. Within the time interval $[t+1; t+\widehat{T}]$, where the period of time $\widehat{T} = O\left(\frac{d \log n}{\mu \max\{(2(c-c_0)d^+ + 1), d^+/2 + 1\}}\right)$ follows from Lemma 6.1, every node with a load of more than cd^+ at time t, will decrease its load below $\bar{x} + \delta d^+ + 2d^\circ + \frac{1}{2} + \frac{1}{2}\max\{(2(c - c_0)d^+), d^+/2\} \leq c_0 d^+ - d^+/2 + \frac{1}{2} + \max\{(d^+(c - c_0)), d^+/4\} \leq cd^+ + \frac{1}{2}$ (and so also below cd^+) at some moment of time during the considered time interval. By Lemma 6.3 a potential drop occurs for $\phi(c)$ in the considered interval $[t+1; t+\widehat{T}]$. More precisely, every node u with $x_t(u) \in [cd^+, cd^+ + s]$ contributes $x_t(u) - cd^+$ to both the potential drop and the value of $s_t(c)$, whereas every node u with $x_t(u) > cd^+ + s$ contributes exactly s to the potential drop and at most d^+ to the value of $s_t(c)$. Hence, we obtain from Lemma 6.3 (and the fact that $s \leq d^+$):

$$\phi_{t+\widehat{T}}(c) \leq \phi_t(c) - \frac{s}{d^+} \cdot s_t(c). \tag{15}$$

Combining (14) and (15), we obtain for any time moment $t \in [T_{p-1}, T_p]$:

$$\phi_{t+\widehat{T}}(c) \leq \phi_t(c) - \frac{s}{d^+} \cdot (\phi_t(c) - \tfrac{b}{2}). \tag{16}$$

We can transform this expression to the following form:

$$\phi_{t+\widehat{T}}(c) - \tfrac{b}{2} \leq \phi_t(c) - \tfrac{b}{2} - \frac{s}{d^+} \cdot (\phi_t(c) - \tfrac{b}{2}) = \left(1 - \frac{s}{d^+}\right) \cdot (\phi_t(c) - \tfrac{b}{2}).$$

Observe that we can fix t_p satisfying (13) so that $t_p \geq \frac{d^+}{s}\widehat{T}$ (which implies $T_p \geq T_{p-1} + \frac{d^+}{s}\widehat{T}$) where we took into account that $2(c - c_0) \geq 2(c_1 - c_0) - p$ for $c \geq c_1 - p/2$. Now, taking advantage of the monotonicity of potentials, we have from (16):

$$\phi_{T_p} \leq \phi_{(T_{p-1} + \frac{d^+}{s}\widehat{T})}(c) \leq \tfrac{b}{2} + \left(1 - \frac{s}{d^+}\right)^{\frac{d^+}{s}} \cdot (\phi_{T_{p-1}}(c) - \tfrac{b}{2}) \leq \tfrac{b}{2} + \tfrac{1}{2}(2b - \tfrac{b}{2}) = \tfrac{3}{4}b \leq b,$$

which completes the inductive proof of the bound on t_p for $c \geq c_1 - p/2$.

Moreover, for $c < c_1 - p/2$, (11) holds because $4^{(c_1 - c)}2^{-p} > 1$, and $\phi_{T_p}(c) \leq (c_1 - c_0)d^+n$ holds by the definition of potentials.

Now, taking into account (12) and (13), we can bound the time of termination of phase p_f of the process as follows:

$$\begin{aligned}
T_{p_f} &= T + \sum_{p=1}^{p_f} t_{p_f} \\
&= T + \sum_{p=1}^{p_f} O\left(\frac{d^+}{s} \frac{d \log n}{\mu \cdot \max\{(2(c_1 - c_0) - p)d^+ + 1, d^+/2 + 1\}}\right) \\
&= O\left(T + \frac{d}{s}\frac{\log n}{\mu}\left(\left(\sum_{p=1}^{2(c_1 - c_0)-1} \frac{1}{2(c_1 - c_0) - p + 1/d^+}\right) + \frac{d^+}{d^+/2 + 1} \cdot \log\left((c_1 - c_0)d^+n\right)\right)\right) \\
&= O\left(T + \frac{d}{s}\frac{\log^2 n}{\mu}\right),
\end{aligned}$$

where we recall that p_f was given by expression (12), and that $c_1 d^+ - c_0 d^+ = O\left(\frac{d \log n}{\mu}\right)$.

In this way, we have shown that a balancedness of $(\delta + 1/2)d^+ + 2d^\circ$ is achieved in time $O\left(T + \frac{d}{s}\frac{\log^2 n}{\mu}\right)$. By using the same technics and Lemma 6.5 instead of Lemma 6.3, we can show that no node has a load of less than $\bar{x} - (\delta + 1/2)d^+ + 2d^\circ$. This gives the desired discrepancy bound of $(2\delta + 1)d^+ + 4d^\circ$. □

The following proposition shows that that strictly δ-fair algorithms are a subclass of cumulatively δ-fair algorithms with a remainder larger that zero.

Proposition 6.6. *For any strictly δ-fair load-balancing algorithm A, there exists a cumulatively δ-fair load-balancing algorithm A', such that for all time steps t and and edges (u, v) we have that the load sent over (u, v) is the same in A and A'.*

Proof. The reformulation of algorithm A as algorithm A' proceeds as follows. For all edges $e \in E(G)$ (*i.e.*, except for self-loops), in every step A' places the same amount of load on e as A. However, in A' load may be retained on nodes in a different way, being placed in the remainder $r_t(u)$ rather than on self-loops at node $u \in V$. To prove, that it is always possible, we proceed by induction.

Specifically, at a fixed moment of time t, let $f_t(e)$ be the amount of load put on an edge e by algorithm A, and $f'_t(e)$ be the amount of load put on an edge by A', and let $F_t(e)$ and $F'_t(e)$ be the respective cumulative loads for algorithms A and A'. Algorithm A' processes all edges (including self-loops) sequentially and verifies if sending this amount of load along e would satisfy the cumulative fairness condition up to time t with respect to all edges outgoing from u already processed.

Let e_1 be the edge or self-loop that violates the cumulative load property for A', that is there exists an incident edge or self-loop e_2 such that $|(F'_{t-1}(e_1) + f_t(e_1)) - (F'_{t-1}(e_2) + f_t(e_2))| > \delta$. Since $|f_t(e_1) - f_t(e_2)| \leq 1$, and $|F'_{t-1}(e_1) - F'_{t-1}(e_2)| \leq \delta$ (from inductive assumption), we get that

$$(F'_{t-1}(e_1) + f_t(e_1)) - (F'_{t-1}(e_2) + f_t(e_2)) \in \{\delta + 1, -\delta - 1\} \tag{17}$$

(without loss of generality we can assume that this value is $\delta + 1$). Moreover, we can show that for every e'_2 such that the pair e_1, e'_2 violates cumulative fairness, the value (17) is $\delta + 1$ (otherwise $F'_{t-1}(e_1) - F'_{t-1}(e_2) = \delta$ and $F'_{t-1}(e_1) - F'_{t-1}(e'_2) = -\delta$ imply $F'_{t-1}(e'_2) - F'_{t-1}(e_2) = 2\delta$ which contradicts the inductive assumption). We can also observe that e_1 is a loop (attached to vertex u), since non-loop edges satisfy cumulative fairness for A. Thus it is enough to set $f'_t(e_1) = f_t(e_1) - 1$ and increase $r_t(u)$ by one (in the mirror scenario with a value of $-\delta - 1$ in (17) we would set $f'_t(e_1) = f_t(e_1) + 1$ and decrease $r_t(u)$ by one). It is easy to observe that this makes e_1 satisfy δ-fairness with every other edge incident to u. After processing all edges and self-loops and edges in this way, we eventually obtain that cumulative fairness is preserved, and moreover $|r'_t(u)| \leq d^\circ \leq d^+$. □

7 Lower Bounds

We start by showing that the cumulative fairness bounds we introduce cannot be completely discarded when improving upon the discrepancy gaps from [16]. Recall that an algorithm is *round-fair* if the load which any node u sends over its edges is either $\left\lfloor \frac{x_t(u)}{d} \right\rfloor$ or $\left\lceil \frac{x_t(u)}{d} \right\rceil$. Note that a round-fair algorithm is not necessary cumulatively δ-fair for any constant δ. In the following we show that there are round-fair algorithms which have a discrepancy of at least $(c \operatorname{diam}(G) \cdot d)$ for some constant c.

Theorem 7.1. *Let G be a d-regular graph. There exists an initial distribution of tokens and a round-fair algorithm A, such that A cannot achieve a discrepancy better than $(c \operatorname{diam}(G) \cdot d)$, for some absolute constant $c > 0$.*

Proof. We will describe an initial state, corresponding to a steady state distribution, such that the flow of load along each edge e is the same in every moment of time ($f_0(e) = f_1(e) = f_2(e) = \ldots$), and the load of nodes does not change in time (however, the load of two distant nodes in the graph will be sufficiently far apart). We take two vertices u and w such that distance between them is $\operatorname{diam}(G)$. We assign to every node $v \in V$ a value $b(v)$ being the shortest path distance from v to u ($b(u) = 0$, $b(v) = 1$ for direct neighbours of u, etc.). For any given edge (v_1, v_2), we assign

$$f_0(v_1, v_2) = \min(b(v_1), b(v_2))$$

We observe, that for each v:

$$\max_{e_1, e_2 \in E_v} |f_0(e_1) - f_0(e_2)| \leq 1$$

and for each edge (v_1, v_2):

$$f_0(v_1, v_2) = f_0(v_2, v_1).$$

Thus, at each step there exists a way to assign values of $\lceil f(v) \rceil$ and $\lfloor f(v) \rfloor$ so as to achieve desired values over edges, and that the system is in the steady state. The sought value of discrepancy is achieved for the considered pair of nodes u and w whose distance in G is $\operatorname{diam}(G)$. □

The following bound shows that the stateless algorithms we design are asymptotically the best possible in terms of eventual discrepancy. Namely, any stateless algorithm is not able to achieve a discrepancy better than cd, for some constant c. This also means that the bound on the discrepancy, presented in Theorem 3.4, cannot be improved in general for the class of strictly fair algorithms.

Theorem 7.2. *Let A be an arbitrary deterministic and stateless algorithm. For every even n, there exists a d-regular graph and an initial load distribution such that A cannot achieve discrepancy better than cd, for some absolute constant $c > 0$.*

Proof. We take an arbitrary graph G with n vertices which contains a $\lfloor d/2 \rfloor$-clique C. We can construct such a graph by taking nodes numbered from 0 to $n-1$ and connecting each pair of nodes i and j with an edge if and only if $(i-j) \bmod n \in \{n - \lfloor d/2 \rfloor, \ldots, n-1, 0, 1, \ldots, \lfloor d/2 \rfloor\}$. If d is odd, we also add edges (i,j) for all $(i-j) \bmod n = n/2$. W.l.o.g. we can assume that $C = \{0, 1, \ldots, \lfloor d/2 \rfloor - 1\}$.

Let A be a deterministic and memoryless algorithm. Since A is deterministic and stateless, for each node u, at round t the load which A sends to neighbours and the load it keeps through the remainder vector depend solely on the current load of u. Let us fix $\ell = |C| - 1$. Initially, the load is distributed in such a way that every node in C has ℓ load and every other node has load 0. For any given node whose current load is ℓ, let p° denote the number of tokens kept by A at the considered node, and let $p_1, p_2, \ldots, p_d$ be the number of tokens sent by A along respective outgoing edges. Clearly, $\ell = p^\circ + \sum_{i=1}^{d} p_i$. At most ℓ of those values are positive, so we can assume w.l.o.g. that $p_d = p_{d-1} = \ldots = p_{\ell+1} = 0$.

We complete the construction in such a way that at each time step the load over every node is preserved. Let us fix $i \in C$. We design an adversary which has control over which values from $\{p_1, \ldots, p_d\}$ are sent along edges of the clique, and which chooses to send along edges of the clique the possibly nonzero values $p_1, p_2, \ldots, p_\ell$. These values will be assigned to the edges $(i, (i+1) \bmod d), (i, (i+2) \bmod d), \ldots, (i, (i-1) \bmod d)$, respectively. We assign all other values arbitrarily since they are all equal to 0. Thus, we observe that at each step loads of nodes are preserved, since the new load of all nodes having load ℓ at the end of a step is $p^\circ + \sum_{i=1}^{\ell} p_i = \ell$ in the next step. All other nodes in $V \setminus C$ will not receive any tokens and they will remain with load 0. Thus, the load of nodes in the graph does not change over rounds and the load difference of nodes in C and the nodes in $V \setminus C$ is cd, for some constant $c > 0$. □

Our final lower bounds concern variants of the ROTOR-ROUTER. The next theorem shows that for a graph without self-loops (*i.e.*, $G = G^+$) the best possible discrepancy of the ROTOR-ROUTER is at least $c \cdot d \cdot \varphi'(G)$, where $\varphi'(G)$ is the odd girth of graph G, *i.e.*, the length of the shortest odd length cycle over all nodes of G. This gives for an odd-length cycle a discrepancy of at least $c \cdot d \cdot \mathrm{diam}$ for some constant c.

Theorem 7.3. *Let G be any d-regular and non-bipartite graph, and let $d^+ = d$. Then, there exists an initial load distribution and direction of the rotors such that* ROTOR-ROUTER *cannot achieve discrepancy better than $(c \cdot d\varphi(G))$, for some absolute constant $c > 0$, where $(2\varphi(G) + 1)$ is the odd girth of G.*

Proof. Let u be an arbitrary vertex belonging to the shortest odd cycle. We assign to every node $v \in V$ a value $b(v)$ being the shortest path distance from v to u ($b(u) = 0$, $b(v) = 1$ for direct neighbours of u, etc.). Observe that for any edge (v_1, v_2), we have $b(v_1) - b(v_2) \in \{-1, 0, 1\}$. Moreover, we have $b(v_1) = b(v_2)$ only if $b(v_1) \geq \varphi(G)$. Suppose $b(v_1) < \varphi(G)$. We can construct an odd length cycle $u \rightsquigarrow v_1 \to v_2 \rightsquigarrow u$ of at most $2 \cdot \varphi(G) - 1$, a contradiction.

For the construction, fix a sufficiently large integer $L > 0$ which will be linked to the average load in the system (it has no effect over the final discrepancy, we need L to be large enough to have nonnegative values of load). We will design a configuration of load in the system which will alternate between two different states, identical for all configurations in odd time steps and even time steps, respectively (thus, $f_0(e) = f_2(e) = \ldots$ and $f_1(e) = f_3(e) = \ldots$). We will describe a configuration at any time step by

providing values distributed over every edge. The load distributed over edge (v_1, v_2) will only depend on the values of $b(v_1)$ and $b(v_2)$. If $b(v_1) \geq \varphi(G)$ or $b(v_2) \geq \varphi(G)$, we set $f_0(v_1, v_2) = L$, otherwise:

$$f_0(v_1, v_2) = \begin{cases} L + (\varphi(G) - \min(b(v_1), b(v_2))) & \text{if } 2|b(v_1) \text{ and } 2 \not| b(v_2), \\ L - (\varphi(G) - \min(b(v_1), b(v_2))) & \text{if } 2 \not| b(v_1) \text{ and } 2|b(v_2). \end{cases}$$

We also set:

$$f_1(v_1, v_2) = f_0(v_2, v_1). \tag{18}$$

Setting all of $f_0(e)$ and $f_1(e)$ is enough to describe every value over every edge. We now prove that such a configuration is possible for some execution of the ROTOR-ROUTER algorithm, *i.e.*, that there exists an ordering of the outgoing edges of each node in the cycle of the ROTOR-ROUTER which leads to such alternating configurations. We observe that $f_t(v_1, v_2) + f_t(v_2, v_1) = 2L$ for $t \in \mathbf{N}$. Thus, for $t \in \mathbf{N}$ we have

$$f_t(v_1, v_2) + f_{t+1}(v_1, v_2) = 2L. \tag{19}$$

We observe, that for any node v and its two neighbours v_1, v_2, it holds

$$|f_t(v, v_1) - f_t(v, v_2)| \leq 1$$

Also, due to (19):

$$\ldots = f_t(v, v_1) - f_t(v, v_2) = -\left(f_{t+1}(v, v_1) - f_{t+1}(v, v_2)\right) = f_{t+2}(v, v_1) - f_{t+2}(v, v_2) = \ldots$$

By (18), the incoming and outgoing flows of load through edges are preserved, and because the difference between outgoing flows is alternating in signs (for two incident outgoing edges), it is always possible to choose an edge ordering in the cycle of the ROTOR-ROUTER representing such a situation. Indeed, we observe that for particular vertex v, outgoing directed edges of v can be partitioned into two sets $P_1 \cup P_2$, where in even steps edges from P_1 are given one more token than edges from P_2, and in odd steps edges from P_1 are given one less token than edges from P_2. So it is enough to select an ordering of edges for the ROTOR-ROUTER such that every edge from P_1 precedes every edge from P_2 set.

We observe that the node u alternates between loads $(L + \varphi(G)) \cdot d$ and $(L - \varphi(G)) \cdot d$, while average load of a node in this setting is exactly $L \cdot d$, which gives us the claimed discrepancy. □

References

[1] C. Adolphs and P. Berenbrink. Distributed self-prefering load balancing with weights and speeds. In *PODC*, pages 135–144, 2012.

[2] C. Adolphs and P. Berenbrink. Improved bounds on diffusion load balancing. In *IPDPS*, 2012.

[3] H. Akbari and P. Berenbrink. Parallel rotor walks on finite graphs and applications in discrete load balancing. In *SPAA*, pages 186–195, 2013.

[4] 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.

[5] P. Berenbrink, C. Cooper, T. Friedetzky, T. Friedrich, and T. Sauerwald. Randomized diffusion for indivisible loads. In *SODA*, pages 429–439, 2011.

[6] J. Cooper, B. Doerr, T. Friedrich, and J. Spencer. Deterministic random walks on regular trees. *Random Struct. Algorithms*, 37(3):353–366, 2010.

[7] J. Cooper and J. Spencer Simulating a random walk with constant error *Combinatorics, Probability and Computing*, 15:815–822, 2006.

[8] B. Doerr and T. Friedrich. Deterministic random walks on the two-dimensional grid. *Combinatorics, Probability and Computing*, 18(1-2):123–144, 2009.

[9] T. Friedrich, M. Gairing, and T. Sauerwald. Quasirandom load balancing. In *SODA*, pages 1620–1629, 2010.

[10] T. Friedrich and T. Sauerwald. Near-perfect load balancing by randomized rounding. In *STOC*, pages 121–130, 2009.

[11] T. Friedrich and T. Sauerwald. The cover time of deterministic random walks. In *COCOON*, pages 130–139, 2010.

[12] S. Kijima, K. Koga, and K. Makino. Deterministic random walks on finite graphs. In *ANALCO*, pages 16–25, 2012.

[13] A. Kosowski, and D. Pająk. Does Adding More Agents Make a Difference? A Case Study of Cover Time for the Rotor-Router. In *ICALP*, pages 544–555, 2014.

[14] D. Levin, Y. Peres, and E. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[15] V. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy. Eulerian walkers as a model of self-organized criticality. *Phys. Rev. Lett.*, 77:5079–5082, 1996.

[16] 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, 1998.

[17] T. Sauerwald and H. Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *FOCS*, pages 341–350, 2012. A revised and extended version is available online at: http://arxiv.org/abs/1201.2715.

[18] R. Subramanian and I. Scherson. An analysis of diffusive load-balancing. In *SPAA*, pages 220–225, 1994.

[19] T. Shiraga, Y. Yamauchi, S. Kijimam, and M. Yamashita. Deterministic Random Walks for Rapidly Mixing Chains. arXiv:1311.3749, 2013.

[20] V. Yanovski, I. Wagner, and A. Bruckstein A Distributed Ant Algorithm for Efficiently Patrolling a Network *Algorithmica*, 37(3):165-186, 2003.

APPENDIX

8 Auxiliary Results

The following technical lemma giving bounds on the behaviour of the error matrix $\mathbf{\Lambda}_\tau$ for $\tau \in \mathbf{N}$ with $\tau \geq t_{mix}$. We recall that $\mathbf{\Lambda}_\tau$ measures the difference between the steady-state distribution and the transition matrix after τ steps, *i.e.*, P^τ.

Lemma 8.1 ([14]). *Let $(\mathbf{q}_t)$ be a sequence of vectors parameterized by t, let $\bar{\mathbf{q}}_t = \mathbf{P}^\infty \mathbf{q}_t$, and let $c \in \mathbf{N}$ be an arbitrary constant.*

(i) For $t \geq c \cdot 4\frac{\log(n \cdot \max_\tau \|\mathbf{q}_\tau - \bar{\mathbf{q}}_\tau\|_\infty)}{\mu}$ we have
$$\|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty \leq 2^{-c}$$

(ii) Let t_{mix} be the mixing time, $t_{mix} = O\left(\frac{\log n}{\mu}\right)$, then
$$\sum_{t \geq c \cdot t_{mix}} \|\mathbf{\Lambda}_t \mathbf{q}_t\|_\infty \leq 2^{-c+1} \cdot \max_{\tau \geq c \cdot t_{mix}} \{\|\mathbf{q}_\tau\|_\infty\}$$