



Distributing Unit Size Workload Packages in Heterogeneous Networks

Robert Elsässer Burkhard Monien Stefan Schamberger

Institute for Computer Science
University of Paderborn
Frstenallee 11, D-33102 Paderborn
{elsa,bm,schaum}@uni-paderborn.de

Abstract

The task of balancing dynamically generated work load occurs in a wide range of parallel and distributed applications. Diffusion based schemes, which belong to the class of nearest neighbor load balancing algorithms, are a popular way to address this problem. Originally created to equalize the amount of arbitrarily divisible load among the nodes of a static and homogeneous network, they have been generalized to heterogeneous topologies. Additionally, some simple diffusion algorithms have been adapted to work in dynamic networks as well. However, if the load is not divisible arbitrarily but consists of indivisible unit size tokens, diffusion schemes are not able to balance the load properly. In this paper we consider the problem of balancing indivisible unit size tokens on heterogeneous systems. By modifying a randomized strategy invented for homogeneous systems, we can achieve an asymptotically minimal expected overload in l_1 , l_2 and l_∞ norm while only slightly increasing the run-time by a logarithmic factor. Our experiments show that this additional factor is usually not required in applications.

Article Type	Communicated by	Submitted	Revised
regular paper	Tomasz Radzik	September 2004	April 2005

This work was partly supported by the German Research Foundation (DFG) project SFB-376 and by the IST Program of the EU under contract numbers IST-1999-14186 (ALCOM-FT) and IST-2001-33116 (FLAGS). Parts of the results have been presented at the European Symposium on Algorithms 2004 [10].

1 Introduction

Load balancing is a very important prerequisite for the efficient use of parallel computers. Many distributed applications produce work load dynamically which often leads to dramatical differences in runtime. Thus, in order to achieve an efficient use of the processor network, the amount of work has to be balanced during the computation. Obviously, we can ensure an overall benefit only if the balancing scheme itself is highly efficient.

If load is arbitrarily divisible, the balancing problem for a homogeneous network with n nodes can be described as follows. At the beginning, each node i contains some work load w_i . The goal is to obtain the balanced work load $\bar{w} = \sum_{i=1}^n w_i/n$ on all nodes. We assume for now that no load is generated or consumed during the balancing process, i.e., we consider a static load balancing scenario.

A popular class of load balancing algorithms consists of diffusion schemes e.g. [5]. They work iteratively and each node migrates a load fraction (flow) over the topology's communication links (edges), depending on the work load difference to its neighbors. Hence, these schemes operate locally and therefore avoid expensive global communication. Formally, if we denote the load of node i after k iteration steps by w_i^k and the flow over edge $\{i, j\}$ in step k by $y_{i,j}^{k-1}$, then

$$\begin{aligned} \forall e = \{i, j\} \in E : y_{i,j}^{k-1} &= \alpha_{i,j}(w_i^{k-1} - w_j^{k-1}); \\ \text{and } w_i^k &= w_i^{k-1} - \sum_{\{i,j\} \in E} y_{i,j}^{k-1}, \end{aligned} \quad (1)$$

is computed where all $\alpha_{i,j}$ satisfy the conditions described in the next Section. Equation 1 can be rewritten in matrix notation as $w^k = Mw^{k-1}$, where the matrix M is called the *Diffusion Matrix* of the network. This algorithm is called *First Order Scheme* (FOS) and converges to a balanced state computing the l_2 -minimal flow. Improvements of FOS are the *Second Order Schemes* (SOS) which perform faster than FOS by almost a quadratic factor [22, 6].

In [8], these schemes are extended to heterogeneous networks consisting of processors with different computing powers. In such an environment, computations perform faster if the load is balanced proportionally to the nodes' computing speed s_i :

$$\bar{w}_i := \frac{\sum_{j=1}^n w_j}{\sum_{j=1}^n s_j} s_i. \quad (2)$$

Obviously, this is a generalization of the load balancing problem in homogeneous networks ($s_i = 1$). Diffusion in networks with communication links of different capacities are analyzed e.g. in [6, 26]. It is shown that the existing balancing schemes can be generalized, such that roughly speaking faster communication links get a higher load migration volume than slower ones. The two generalizations can be combined as described in [8]. Heterogeneous topologies are extremely attractive because they often appear in real hardware installations

containing machines and networks of different capabilities. In [9] it is demonstrated that FOS is also applicable to balance load in dynamic networks where edges fail from time to time or are present depending on the distance of the moving nodes.

In contrast to the above implementations, work load in real world applications usually cannot be divided arbitrarily often, but only to some extent. Thus, to address the load balancing problem properly, a more restrictive model is needed. The unit-size token model [22] assumes a smallest load entity, the *unit-size token*. Furthermore, work load is always represented by a multiple of this smallest entity. It is clear that load in this model is usually not completely balanceable. Unfortunately, as shown for FOS in homogeneous networks in e.g. [22, 23, 7], the use of integer values prevents the known diffusion schemes to balance the load completely. Especially if only a relatively small amount of tokens exists, a considerable load difference remains.

To measure the balance quality in a network, usually two metrics are considered. The l_2 -norm expresses how well the load is distributed on the processors, hence the error in this norm $\|e\|_2 = \sqrt{\sum_{i=1}^n (w_i^k - \bar{w}_i)^2}$ should be minimized. One is even more interested in minimizing the overall computation time and therefore the load in l_∞ -norm, which is equivalent to minimizing the maximal (weighted) load occurring on the processors. Recall, that if the error $\|e\|_\infty = \max |\bar{w} - w_i|$ is less than b , then $\|e\|_2$ is bounded by $\sqrt{n} \cdot b$.

The problem of balancing indivisible tokens in homogeneous networks is addressed e.g. in [3]. The proposed algorithm improves the local balance significantly, but still cannot guarantee a satisfactory global balance. The authors of [22] use a discretization of FOS by transmitting the flow $y_{i,j} = \lfloor \alpha_{i,j} (w_i^{k-1} - w_j^{k-1}) \rfloor$ over edge $\{i, j\}$ in step k . After $k = \Theta((d/\lambda_2) \log(En))$ steps, the error in l_2 -norm is reduced to $\mathcal{O}(nd^2/\lambda_2)$, where $E = \max_i |w_i^0 - \bar{w}_i|$ is the *maximal initial load imbalance* in the network, λ_2 denotes the second smallest eigenvalue of the Laplacian of the graph (see next Section), and d is the maximal vertex degree. This bound has been improved by showing that the error in l_∞ -norm is $\mathcal{O}((d^2/\lambda_2) \log(n))$ after $k = \Theta((d/\lambda_2) \log(En))$ iterations. [23]. Furthermore, there exist a number of results to improve the balance on special topologies, e.g. [13, 18, 17]. The randomized algorithm developed in [7] reduces $\|e^k\|_2$ to $\mathcal{O}(\sqrt{n})$ in $k = \Theta((d/\lambda_2) (\log(E) + \log(n) \log(\log(n))))$ iteration steps with high probability, if \bar{w}_i exceeds some certain threshold. Since it is also shown that in l_2 -norm $\mathcal{O}(\sqrt{n})$ is the best expected asymptotic result that can be achieved concerning the final load distribution [7], this algorithm provides an asymptotic optimal result. However, concerning the minimization of the overall computation time, the above algorithm only guarantees $\|w^k\|_\infty \leq \bar{w}_i + \mathcal{O}(\log(n))$ within the previously described number of steps k . Since $\|w^k\|_\infty = \bar{w}_i + \Omega(\log(n)/\log(\log(n)))$ can occur, this algorithm does not achieve an optimal situation concerning the maximal overload in the system.

In this paper, we present a modification of the randomized algorithm from [7] in order to minimize the overload in the system in l_1 , l_2 - and l_∞ -norm, respectively. We show that this algorithm can also be applied in heterogeneous

networks, achieving the corresponding asymptotic overload for the weighted l_1 -, l_2 -, or l_∞ -norms (for a definition refer to Section 3). The algorithm’s run-time increases slightly by at most a logarithmic factor and is bounded by $\Theta((d/\lambda_2) \cdot (\log(E) + \log^2(n)))$ with high probability, whenever \bar{w}_i exceeds some threshold. Note, that from our experiments we can conclude that the additional run-time is usually not needed. Furthermore, our results can be generalized to dynamic networks which are modeled as graphs with a constant node set, but a varying set of edges during the iterations. Similar dynamic networks have been used to analyze the token distribution problem in [14]. However, in contrast to diffusion, the latter only allows one single token to be sent over an edge in each iteration.

The outline of the paper is as follows. In the next section, we give an overview of the basic definitions and the necessary theoretical background. Then, we describe the randomized strategy for heterogeneous networks and show that this algorithm minimizes the overload to the asymptotically best expected value whenever the heterogeneity obeys some restrictions. In Section 4, we present some experimental results and finally Section 5 contains our conclusion.

2 Background and Definitions

Let $G = (V, E)$ be an undirected, connected, weighted graph with $|V| = n$ nodes and $|E| = m$ edges. Let $c_{i,j} \in \mathbb{R}$ be the *capacity* of edge $e_{i,j} \in E$, s_i be the *processing speed* of node $i \in V$, and $w_i \in \mathbb{R}$ be its *work load*. In case of indivisible load entities, this value represents the number of unit-size tokens.

Let $A \in \mathbb{R}^{n \times n}$ be the *Weighted Adjacency Matrix* of G . As G is undirected, A is symmetric. Column/row i of A contains $c_{i,j}$ where j and i are neighbors in G . For some of our constructions we need the *Laplacian* $L \in \mathbb{Z}^{n \times n}$ of G defined as $L := D - A$, where $D \in \mathbb{N}^{n \times n}$ contains the weighted degrees as diagonal entries, e.g. $D_{i,i} = \sum_{\{i,j\} \in E} c_{i,j}$, and 0 elsewhere.

The Laplacian L and its eigenvalues are used to analyze the behavior of diffusion in homogeneous networks. For heterogeneous networks we have to apply the generalized Laplacian LS^{-1} , where $S \in \mathbb{R}^{n \times n}$ denotes the diagonal matrix containing the processor speeds s_i (cf. [8]). We assume that $1 \leq s_i \leq \mathcal{O}(n^\delta)$ with $\delta < 1$.

Generalizing the local iterative algorithm of equation (1) in the case of arbitrary divisible tokens to heterogeneous networks, one yields [8]:

$$w_i^k = w_i^{k-1} - \sum_{j \in N(i)} c_{i,j} \left(\frac{w_i^{k-1}}{s_i} - \frac{w_j^{k-1}}{s_j} \right) \quad (3)$$

Here, $N(i)$ denotes the set of neighbors of $i \in V$, and w_i^k is the load of the node i after the k -th iteration. As mentioned, this scheme is known as FOS and converges to the average load \bar{w} of equation (2) [8]. It can be written in matrix notation as $w^k = Mw^{k-1}$ with $M = I - LS^{-1} \in \mathbb{R}^{n \times n}$. M contains $c_{i,j}/s_j$ at position (i, j) for every edge $e = \{i, j\}$, $1 - \sum_{e=\{i,j\} \in E} c_{i,j}/s_j$ at diagonal entry i , and 0 elsewhere.

Now, let λ_i , $1 \leq i \leq n$ be the eigenvalues of the Laplacian LS^{-1} in increasing order. It is known that $\lambda_1 = 0$ with (right) eigenvector (s_1, \dots, s_n) [4]. The values of $c_{i,j}$ have to be chosen such that the diffusion matrix M has the eigenvalues $-1 < \mu_i = 1 - \lambda_i \leq 1$. In the rest of the paper, we assume that $c_{i,j} = 1/(c \cdot \max\{d_i, d_j\})$ if i and j share an edge, and $c_{i,j} = 0$ otherwise, where d_i is the degree of node i and $c \in (1, 2]$ is a constant. Note that by using this choice for $c_{i,j}$ the nodes do not need to have any global knowledge about the network. Since G is connected, the first eigenvalue $\mu_1 = 1$ is simple with eigenvector (s_1, \dots, s_n) . We denote by $\gamma = \max\{|\mu_2|, |\mu_n|\} < 1$ the second largest eigenvalue of M according to absolute values and call it the *diffusion norm* of M . Certainly, γ governs the convergence of the process described in equation (3). Choosing $c_{i,j}$ as described above, $1/(1 - |\mu_n|) = \mathcal{O}(1)$. Therefore, in the case when $\gamma = |\mu_n|$, $1/(1 - |\mu_n|)$ and $1/(1 - |\mu_2|)$ differ only in a constant factor, and we still can express the asymptotic behaviour of the scheme by μ_2 . Otherwise, the convergence is described by μ_2 . Hence, we can ignore μ_n here (see [9]).

Several modifications to FOS have been discussed in the past. One of them is SOS [22], which has the form

$$w^1 = Mw^0, w^k = \beta Mw^{k-1} + (1 - \beta)w^{k-2}, \quad k = 2, 3, \dots \quad (4)$$

Setting $\beta = 2/(1 + \sqrt{1 - \mu_2^2})$ results in the fastest convergence.

As described in the introduction, we denote the *error* after k iteration steps by e^k , where $e^k = w^k - \bar{w}$. The convergence rate of diffusion schemes in the case of arbitrary divisible tokens depends on how fast the system becomes ϵ -balanced, i.e., the final error $\|e^k\|_2$ is less than $\epsilon\|e^0\|_2$. By using simple calculations and the results from [22, 8, 9], we obtain the following lemma.

Lemma 1 *Let G be a graph and L its Laplacian, where $c_{i,j} = 1/(c \cdot \max\{d_i, d_j\})$, d_i is the degree of node i and $c \in (1, 2]$ a constant. Let $M = I - LS^{-1}$ be the diffusion matrix of G and set $\beta = 2/(1 + \sqrt{1 - \gamma^2})$. Then, FOS and SOS require $\Theta((d/\lambda_2) \cdot \ln(s_{\max}/(\epsilon s_{\min})))$ and $\Theta(\sqrt{d/\lambda_2} \cdot \ln(s_{\max}/(\epsilon s_{\min})))$ steps, respectively, to ϵ -balance the system, where $s_{\max} = \max_i s_i$ and $s_{\min} = \min_i s_i$.*

Although SOS converges faster than FOS by almost a quadratic factor and therefore seems preferable, it has a drawback. During the iterations, the outgoing flow of a node i may exceed w_i which results in negative load. On static networks, the two phase model copes with this problem [6]. The first phase determines the balancing flow while the second phase then migrates the load accordingly. However, this two phase model cannot be applied on dynamic networks because edges included in the flow computation might not be available for migration.

Even in cases where the nodes' outgoing flow does not exceed their load, we have not been able to show the convergence of SOS in dynamic systems yet. Hence, our diffusion scheme analysis on such networks is restricted to FOS.

Since the diffusion matrix M is stochastic, FOS can also be interpreted as a Markov process, where M is the corresponding transition matrix and the bal-

anced load situation is the related stationary distribution¹. Using this Markov chain approach, the following lemma can be stated [12, 21].

Lemma 2 *Let G be a graph and we assume that a token performs a random walk on G according to the diffusion matrix M as defined in Lemma 1. If the values s_i are in a range $[1, n^\delta]$, $\delta \in [0, 1)$, then for any constant τ , a constant a exists such that after $a((d \cdot \log(n))/\lambda_2)$ steps the probability p_i of the token being situated on some node i is bounded by $(s_i/\sum_{j=1}^n s_j) - (1/n^\tau) \leq p_i \leq (s_i/\sum_{j=1}^n s_j) + (1/n^\tau)$.*

3 The Randomized Algorithm

In this Section, we describe a randomized algorithm that guarantees a final weighted overload of $\mathcal{O}(1)$ in l_∞ -norm for the load balancing problem of indivisible unit-size tokens in heterogeneous networks.

First, we show that FOS as well as SOS do not guarantee a satisfactory balance in case of unit size tokens on inhomogeneous networks. As mentioned, the outgoing load from some node i may exceed w_i when applying SOS. Hence, simply rounding down the flow to the nearest integer does not lead to the desired algorithm. Therefore, the authors of [22] introduced a so called ‘I Owe You’ (IOU) unit on each edge. These IOUs represent the difference between the total flow moved along an edge in the idealized scheme and in the realistic algorithm. If the IOUs accumulate, then we can not prove any result w.r.t. the convergence of the adapted SOS. Nevertheless, in most of the simulations on static networks, even if the IOUs emerge, the number of owed units becomes very small after a few rounds.

If we assume that the IOUs tend to zero after a few iterations, then we can use the techniques of [15] to state the following theorem.

Theorem 1 *Let $G = (V, E)$ be a node weighted graph, where s_i is the weight of node $i \in V$, and let w^0 be the initial load distribution on G . Executing SOS on G , we assume that after a few iteration steps the IOUs are bounded by some constant on each edge of the graph. Then, after sufficient number of steps k , the error $\|e^k\|_2$ is $\mathcal{O}((\sqrt{n} \cdot s_{\max} \cdot d)/(\sqrt{s_{\min}} \cdot (1 - \gamma)))$, where γ is the diffusion norm of the corresponding diffusion matrix M .*

Proof: During the SOS iterations (equation (4)), we round the flow such that only integer load values are shifted over the edges. Hence, we get

$$w^k = \beta M w^{k-1} - (\beta - 1)w^{k-2} + \delta_{k-1}, \quad (5)$$

where δ_k is the round-off error in iteration k . Comparing this to the balancing algorithm with arbitrary divisible load w'^k (equation (4)), by subtracting equation (4) from (5), we get an accumulated round-off

$$a_k = w^k - w'^k = \beta M a^{k-1} - (\beta - 1)a^{k-2} + \delta_{k-1},$$

¹We use M instead of M^T as the transition matrix of the corresponding Markov chain, and consider its right eigenvectors. Accordingly, the stationary distribution corresponds to the right eigenvector of M with eigenvalue 1.

where $a_0 = 0$ and $a_1 = \delta_0$. We know that for any eigenvector z_i of M an eigenvector u_i of $S^{-1/2}LS^{-1/2}$ exists such that $z_i = S^{1/2}u_i$. We also know that the eigenvectors z_i form a basis in \mathbb{R}^n [8]. On the other hand, $a^k = \sum_{i=2}^n \beta_i z_i$ for some $\beta_2, \dots, \beta_n \in \mathbb{R}$ and any $k \in \mathbb{N}$. Combing the techniques from [15] and [8], we therefore obtain

$$\|a_k\|_2 \leq \sum_{j=0}^{k-1} \sqrt{\frac{s_{\max}}{s_{\min}}} (\beta - 1)^{\frac{k-j-1}{2}} (k-j) \delta \leq \sqrt{\frac{s_{\max}}{s_{\min}}} \left(\frac{1-r^{k+1}}{(1-r)^2} - \frac{(k+1)r^k}{1-r} \right) \delta,$$

where δ is chosen such that $\|\delta_j\|_2 \leq \delta$ for any $j \in \{0, \dots, k-1\}$ and $r = \sqrt{\beta-1}$. Since $\|\delta_j\|_2 \leq \sqrt{n} \cdot d$, the theorem follows. \square

Using similar techniques, the same bound can also be obtained for FOS (without the restrictions). Due to classical isoperimetric bounds on the second smallest eigenvalue of the normalized Laplacian of a connected unweighted graph [2], $\lambda_2 = \Omega(1/n^2)$, and since $s_i \leq n$ for any i , it holds that $1/(1-\gamma) = \mathcal{O}(n^3)$. Therefore, the load discrepancy can at most be $\mathcal{O}(n^t)$ with t being small constant. In the rest of the paper we refer to this constant t as the *discrepancy constant*. However, a satisfactory final balance cannot be guaranteed in the case of indivisible unit size tokens.

We now present a randomized strategy, which reduces the final weighted load to an asymptotic optimal value. In the sequel, we only consider the load in l_∞ -norm, although we can also prove that the algorithm described below achieves an asymptotic optimal balance in l_1 - and l_2 -norm.

In order to show the optimality, we have to consider the lower bound on the final weighted load deviation. Let $\tilde{w}_i^k = w_i^k / \bar{s}_i$ be the *weighted load* of node i after k steps, where $\bar{s}_i = n \cdot s_i / (\sum_{j=1}^n s_j)$ is its *normalized processing speed*.

Proposition 1 *There exist node weighted graphs $G = (V, E)$ and initial load distributions w , such that for any load balancing scheme working with indivisible unit-size tokens the expected final weighted load in l_∞ -norm, $\lim_{k \rightarrow \infty} \|\tilde{w}^k\|$, is $\bar{w}_i / \bar{s}_i + \Omega(1)$.*

To see this, let G be a homogeneous network ($s_i = 1$ for $i \in V$) and let $\bar{w}_i = b' + b$, where $b' \in \mathbb{N}$ and $b \in (0, 1)$. Obviously, this fulfills the proposition.

The following randomized algorithm reduces the final load $\|\tilde{w}^k\|_\infty$ to $\bar{w}_i / \bar{s}_i + \mathcal{O}(1)$ in heterogeneous networks. We assume that the weights s_i are lying in the range $[1, n^\delta]$ with $\delta < 1$, and that the number of tokens $\sum_{i=1}^n w_i$ is high enough. Although we mainly describe the algorithm for static networks, it can also be used on dynamic networks, obtaining the same bound on the final load deviation.

The algorithm we propose consists of two main phases. In the first phase, we perform FOS as described in [8] to reduce the load imbalance in l_1 -norm to $\mathcal{O}(n^{t+1})$, where t is the discrepancy value. At the same time, we approximate \bar{w}_i within an error of $\pm 1/n^2$ by using the diffusion algorithm for arbitrarily divisible load. Due to Lemma 1, in this first phase we perform $\mathcal{O}(d \log(nE) / \lambda_2)$ steps. It

is worth mentioning that the nodes do not need to have any global information (excepting n), and we can use the usual termination detection algorithms [11, 24, 27] to determine whether the system has achieved the state described above.

In the second phase, we perform so called *main random walk steps* (MRWSs) described in the following. In a preparation step, we mark the tokens that take part in the random walk and also introduce participating “negative tokens” that represent load deficiencies. Let w_i be the current (integer) load and R be the set of nodes with $\bar{s}_i < 1/3$. If a node $i \in R$ contains less than $\hat{w}_i = \lceil \bar{w}_i \rceil$ tokens, we place $\hat{w}_i - w_i$ “negative tokens” on it. If $i \in R$ owns more than \hat{w}_i tokens, we mark $w_i - \hat{w}_i$ of them. On a node $i \in V \setminus R$ we set $\hat{w}_i = \lceil \bar{w}_i \rceil + \lceil 2\bar{s}_i \rceil$. If such an i has less than \hat{w}_i tokens, we create $\hat{w}_i - w_i$ “negative tokens”. Accordingly, if $i \in V \setminus R$ contains more than \hat{w}_i tokens, we mark $w_i - \hat{w}_i$ of them. Obviously, the number of “negative tokens” is larger than the number of marked tokens by an additive value of $\Omega(n)$. Now, all “negative tokens” and all marked tokens perform random walk steps according to M until at each node the number of tokens is less than \hat{w}_i . Note, that if a “negative token” is moved from node i to node j , a real token is transferred in the opposite direction from node j to node i . Furthermore, if a “negative token” and a marked token meet on a node, they eliminate each other. In the sequel, we call one single step, in which each “negative token” and each marked token performs one random walk step, a *Main Random Walk Step* (MRWS). Again, the usual termination detection algorithms can be used to determine whether the second phase has been completed, and hence the nodes do not need to have any global information.

In the remaining part of this section we prove the correctness of the algorithm and analyze its run-time. In contrast to the above description, we assume for the analysis that “negative tokens” and marked tokens do not eliminate each other instantly, but only after completing $a \ln(n)/\lambda_2$ MRWSs, where a is the constant of Lemma 2 for $\tau = t + 4$ with t being the discrepancy constant. The sequence of $a \ln(n)/\lambda_2$ MRWSs is called in the remaining of this section a *Main Random Walk Round* (MRWR). This modification simplifies the proof, although we will see that the same bounds can also be obtained for the original randomized algorithm.

In order to show the correctness of the algorithm, we first need some auxiliary lemmas.

Lemma 3 *If $p_i \leq \bar{s}_i/n + 1/n^{t+4}$ and $q_j \leq 1 - \bar{s}_i/n + 1/n^{t+4}$, where t is the constant defined above, then it holds that*

$$\binom{m}{k} p_i^k q_j^{m-k} \leq \binom{m}{k} (1/n)^k (1 - 1/n)^{m-k} (1 + O(1/n^{2-\delta})),$$

where $1/n^\delta \leq \bar{s}_i \leq n^\delta$, $k > m\bar{s}_i/n$ and $m \geq n$.

Proof: First we show by induction that $\binom{m}{k} p_i^k \leq \binom{m}{k} \left(\frac{\bar{s}_i}{n}\right)^k \left(1 + \frac{k}{n^{t+3-\delta}} + \frac{k}{n^{t+5-\delta}}\right)$. Obviously, the claim holds for $k = 1$. Let us assume that it also holds for $k - 1$. Then

$$\binom{m}{k} p_i^k = \binom{m}{k-1} p_i^{k-1} \frac{m-k+1}{k} p_i$$

$$\begin{aligned}
 &\leq \binom{m}{k-1} \left(\frac{\bar{s}_i}{n}\right)^{k-1} \left(1 + \frac{k-1}{n^{t+3-\delta}} + \frac{k-1}{n^{t+5-\delta}}\right) \\
 &\quad \cdot \frac{m-k+1}{k} \cdot \frac{\bar{s}_i}{n} \left(1 + \frac{1}{n^{t+3\bar{s}_i}}\right) \\
 &\leq \binom{m}{k} \left(\frac{1}{n}\right)^k \left(1 + \frac{k}{n^{t+3-\delta}} + \frac{k}{n^{t+5-\delta}}\right)
 \end{aligned}$$

On the other hand $(1 - \frac{\bar{s}_i}{n} + \frac{1}{n^{t+4}})^{m-k} \leq (1 - \frac{\bar{s}_i}{n})^{m-k} \left(1 + \frac{2(m-k)}{n^{t+3-\delta}} + \frac{m-k}{n^{t+5-\delta}}\right)$, which can also be shown the same way. The claim holds for $m-k = 1$. Assuming the claim holds for $m-k-1$, we obtain

$$\begin{aligned}
 \left(1 - \frac{\bar{s}_i}{n} + \frac{1}{n^{t+4}}\right)^{m-k} &\leq \left(1 - \frac{\bar{s}_i}{n}\right)^{m-k-1} \left(1 + \frac{2(m-k-1)}{n^{t+3-\delta}} + \frac{m-k-1}{n^{t+5-\delta}}\right) \\
 &\quad \cdot \left(1 - \frac{\bar{s}_i}{n}\right) \left(1 + \frac{2}{n^{t+3-\delta}}\right) \\
 &\leq \left(1 - \frac{\bar{s}_i}{n}\right)^{m-k} \left(1 + \frac{2(m-k)}{n^{t+3-\delta}} + \frac{m-k}{n^{t+5-\delta}}\right)
 \end{aligned}$$

and the lemma follows. \square

Lemma 4 *Let $G = (V, E)$, $|V| = n$ be a node weighted graph. Assume that qn tokens are executing random walks, according to G 's diffusion matrix, of length $(ad \ln(n))/\lambda_2$, $q > 32 \ln(n)$, where a is the same constant as in Lemma 2 for $\tau = t + 4$. Then, a constant c exists, such that, after completion of the random walk, the number of tokens producing overload is less than $cn\sqrt{q \ln(n)}$ with probability $1 - o(1/n)$.*

Proof: We can use the techniques from [7] to show the theorem. Due to lemma 2, after $ad \ln(n)/\lambda_2$ steps each token lies on a certain node i with probability $p_i \in [\bar{s}_i/n - 1/n^{t+4}, \bar{s}_i/n + 1/n^{t+4}]$. Then, the probability that on node i are more than ϕ tokens is

$$P_{i,\phi} \leq \sum_{j=\phi}^{qn} \binom{qn}{j} (\bar{s}_i/n + 1/n^{t+4})^j (1 - \bar{s}_i/n + 1/n^{t+4})^{qn-j}.$$

Due to lemma 3 we get

$$P_{i,\phi} \leq \sum_{j=\phi}^{qn} \binom{qn}{j} (\bar{s}_i/n)^j (1 - \bar{s}_i/n)^{qn-j} (1 + O(1/n^{2-\delta})).$$

For the next, we only consider

$$\tilde{P}_{i,\phi} = \sum_{j=\phi}^{qn} \binom{qn}{j} (\bar{s}_i/n)^j (1 - \bar{s}_i/n)^{qn-j},$$

where $q > 32 \ln(n)$. If $\bar{s}_i q > \ln(n)$, then let $\phi = (1 + \frac{\varphi}{\sqrt{\bar{s}_i q}}) \bar{s}_i q$, where $\varphi \leq c\sqrt{\ln(n)}$ and $c \in (2\sqrt{2}, \leq \sqrt{q/4 \ln(n)})$ is a constant. Combining a Chernoff type bound [1] with the methods from [7], we obtain

$$\tilde{P}_{i,\phi} \leq \left(\frac{1}{e}\right)^{\frac{\varphi^2}{4}(1-o(1))}.$$

If $\bar{s}_i q \leq 32 \ln(n)$, then we can use the Chernoff bound to show that

$$\tilde{P}_{i,(e^2+1)\ln n} \leq o(1/n^2).$$

Using the results above, we know that on some node i with $\bar{s}_i q > \ln(n)$ an overload of at most $c\sqrt{\bar{s}_i q \ln(n)}$ can occur. If $\bar{s}_i q \leq \ln(n)$, then the overload can be at most $(e^2 + 1) \ln(n)$. Applying the Cauchy-Schwarz inequality to the sum of the overloads, we obtain the lemma. \square

Due to the previous lemma, after $\mathcal{O}(\log(\log(n)))$ MRWRs, the maximal load imbalance is $\mathcal{O}(\log(n))$ and the number of tokens producing an overload is $\mathcal{O}(n \log(n))$ (with probability $1 - o(1/n)$).

The next lemma handles the case when the number of tokens producing overload in the system is $\mathcal{O}(n \log(n))$.

Lemma 5 *Let $G = (V, E), |V| = n$ be a node weighted graph and let $q_1 n$ be the number of marked tokens and $q_2 n$ be the number of “negative tokens” which perform random walks of length $(\text{ad} \ln(n))/\lambda_2$ on G , respectively. If $q_1, q_2 = \mathcal{O}(\ln(n))$ with $q_2 n - q_1 n = \Omega(n)$ and $q_1 > e^2 \ln(n)/n$, then a constant $c > 1$ exists, such that, after completion of the random walk, the number of tokens producing overload is less than $\lceil q_1 n/c \rceil$ with probability $1 - o(1/n)$. If $q_1 \leq e^2 \ln(n)/n$, then an arbitrary marked token is destroyed with probability at least $((e - 1)/e) \cdot (1 - o(1))$.*

Proof: First, we describe the distribution of the “negative tokens” on G ’s nodes. Obviously, with probability at least $1/2$ a token is placed on a node i with $\bar{s}_i > 1/2$. Using the Chernoff bound [16], it can be shown that with probability $1 - o(1/n^2)$ at least $q_2 n(1 - o(1))/2$ tokens are situated on nodes with $\bar{s}_i > 1/2$ after a MRWR. With help of the Chernoff bound we can also show that for any node i with $\bar{s}_i \geq c' \ln(n)$ and c' being a proper constant, with probability $1 - o(1/n^2)$ there are at least $q_2 \bar{s}_i/2$ tokens on i after the MRWR. If G contains less than $c' \ln(n)$ nodes with normalized processing speed $c' \ln(n)/2^j < \bar{s}_i < c' \ln(n)/2^{j-1}$ for some $j \in \{1, \dots, \log(c' \ln(n)) + 1\}$, then a token lies on one of these nodes with probability $\mathcal{O}(\ln^2(n)/n)$. Otherwise, if there are $Q > c' \ln(n)$ nodes with speed $c' \ln(n)/2^j < \bar{s}_i < c' \ln(n)/2^{j-1}$, then at least Q/ρ_j of these nodes contain more than $\lceil q_2 c' \ln(n)/2^{j+1} \rceil$ tokens, where $\rho_j = \max\{\lceil 2/(q_2 c' \ln(n)/2^j) \rceil, 2\}$.

Now we turn our attention to the distribution of the marked tokens after an MRWR. Let $q_1 n \geq \sqrt{n}$. Certainly, w.h.p. $q_1 n(1 - o(1))/2$ tokens reside on nodes with $\bar{s}_i > 1/2$. On a node i with normalized processing speed $\bar{s}_i >$

$c' \ln(n)$ are placed at most $q_1 \bar{s}_i(1 + o(1)) + \mathcal{O}(\ln(n))$ tokens. Therefore, most of the marked tokens on such heavy nodes will be eliminated by the “negative tokens”. Let S_j be the set of nodes with $c' \ln(n)/2^j < \bar{s}_i < c' \ln(n)/2^{j-1}$, where $j \in \{1, \dots, \log(c' \ln(n))\}$. We ignore the tokens in sets S_j with $|S_j| \leq \sqrt{n}$, since their number adds up to at most $\mathcal{O}(\sqrt{q_1 n} \ln^2(n))$. Each of the other sets S_j with $|S_j| > \sqrt{n}$, contains $n_S = \Omega(\sqrt{q_1 n})$ marked tokens distributed nearly evenly. Therefore, a constant c'' exists so that at least n_S/c'' of them are eliminated by “negative tokens”. If $q_1 n \leq \sqrt{n}$, then after the MRWR it holds that a marked token lies on some node i with normalized speed $\bar{s}_i > 1/2$ with a probability of at least $1/2$. If one of these tokens is situated on some node i with $\bar{s}_i \geq c' \ln(n)$, then it is destroyed by some “negative token” with high probability. If a marked token resides on some node of the sets S_j , where $j \in \{1, \dots, \log(c' \ln(n))\}$, then a constant $c_f > 0$ exists such that this token is destroyed with probability $1/c_f$. Therefore, a marked token is destroyed with probability of at least $1/(2c_f)$.

Summarizing, the number of marked tokens is reduced to \sqrt{n} after $\mathcal{O}(\ln(n))$ MRWRs. Additional $\mathcal{O}(\ln(n))$ MRWRs decrease this number to at most $\mathcal{O}(\ln(n))$. These can be eliminated within another $\mathcal{O}(\ln(n))$ MRWRs with probability $1 - o(1/n)$. \square

Combining the Lemmas 4 and 5 we obtain the following theorem.

Theorem 2 *Let $G = (V, E)$ be a node weighted graph with maximum vertex degree d and let λ_2 be the second smallest eigenvalue of the weighted Laplacian of G . Furthermore, let w^0 be the initial load and $E = \max_i |w_i^0 - \bar{w}_i|$ the initial maximal load imbalance. If $\sum_{i=1}^n \bar{w}_i$ exceeds some certain threshold, then the randomized algorithm reduces the weighted load in l_∞ -norm, $\|\tilde{w}^k\|_\infty$ to $\bar{w}_i/\bar{s}_i + \mathcal{O}(1)$ in $k = \mathcal{O}((d/\lambda_2)(\log(E) + \log^2(n)))$ iteration steps with probability $1 - o(1/n)$.*

Proof: We know that after the first phase the load imbalance in l_1 -norm is reduced to some value $O(n^{t+1})$, where t is the small discrepancy constant. Moreover, after this phase every node approximates \bar{w}_i within an error of $\pm 1/n^2$, and it also follows from the description of the algorithm that this phase takes time $\mathcal{O}(d \log(n)/\lambda_2)$. For simplicity, we assume from now on that every node i has computed the value \bar{w}_i exactly.

As described in the previous paragraphs, we so far have assumed for the analysis that “negative tokens” and marked tokens do not eliminate each other instantly, but only after completing a MRWR. However, if they eliminate each other immediately, then the number of destroyed tokens after a MRWR is at least half of what we computed in Lemma 4 and Lemma 5.

In order to prove this, we define dead “negative” and marked tokens in the following way. If a “negative token” and a marked token meet each other during the execution of the algorithm, they die without eliminating each other. The dead tokens continue the random walk until an MRWR is finished, and without influencing any other token in the system. After the MRWR has been completed, a situation can occur where a live (marked or “negative”) token cannot be eliminated any more, because its counterparts have been killed before.

However, they are counted as eliminated in the analysis of Lemma 4 and 5. Nevertheless, since each of these uneliminated tokens has a dead counterpart, it holds that the number of eliminated tokens after an MRWR is at least half of what we computed in Lemma 4 and Lemma 5.

Now, applying lemma 4 and 5 together with the arguments described above, we can conclude that after $k = \mathcal{O}((d/\lambda_2)(\log(E) + \log^2(n)))$ steps it holds that $\|\vec{w}^k\|_\infty = \bar{w}_i/\bar{s}_i + \mathcal{O}(1)$. \square

In the previous analysis we assume that the tokens are evenly distributed on the graphs' nodes after each MRWR. However, after the marked and “negative tokens” have eliminated each other, the distribution of the remaining entities no longer corresponds to the stationary distribution of the Markov process. For the analysis, we therefore perform another MRWR to regain an even distribution. Indeed, the tokens are not evenly distributed after the elimination step, but they are close to the stationary distribution in most cases. Hence, the analysis is not tight what can also be seen in the experiments presented in the next section. Using the techniques of [25], we could improve the upper bound for the run-time of the algorithm for special graph classes. However, we do not see how to obtain better run-time bounds in the general case.

Theorem 2 implies that asymptotic optimal results can also be obtained for the weighted overload w.r.t. the l_1 and l_2 -norm. Moreover, the results can also be generalized to dynamic networks (see [9] for the case of arbitrary divisible tokens). Dynamic networks can be modeled by a sequence $(G_i)_{i \geq 0}$ of graphs, where G_i is the graph which occurs in iteration step i [14]. Let us assume that any graph G_i is connected and let $h \geq 1/C \sum_{i=kC+1}^{(k+1)C} (d_{\max}^i/\lambda_2^i)$ for any $k \in \mathbb{N}$, where C is a large constant, d_{\max}^i represents the maximum vertex degree and λ_2^i denotes the second smallest eigenvalue of G_i 's Laplacian. Then, the randomized algorithm reduces $\|\vec{w}^k\|_\infty$ to $\bar{w}_i/\bar{s}_i + \mathcal{O}(1)$ in $k = \mathcal{O}(h(\log(E) + \log^2(n)))$ iteration steps (with probability $1 - o(1/n)$).

4 Experiments

In this section we present some of the experimental results we obtained with our implementation of the approach proposed in the last Section. Our tests are performed in two different environments. The first one is based on common network topologies and therefore mainly considers static settings. However, by letting each graph edge only be present with a certain probability p in each iteration, it is possible to simulate e.g. edge failures and therefore dynamics. The second environment is taken from the mobile ad-hoc network community. Here, nodes move around in a simulation space and a link between two nodes is present if their distance is small enough.

The algorithm works as described in Section 3 and consists of two main phases. First, we balance the tokens as described in [22] only sending the integer amount of the calculated flow. Simultaneously and independently, we simulate the diffusion algorithm with arbitrarily divisible load. Note, that in

Table 1: Graphs and some of their properties used in the experiments.

Graph	V	E	degree			girth	diam.	λ_2
			min	avg	max			
TORUS(16x16)	256	512	4	4.00	4	4	16	0.152241
GRID(16x16)	256	480	2	3.96	4	4	30	0.038429
BFLY(6)	384	768	4	4.00	4	4	9	0.396125
CCC(6)	384	576	3	3.00	3	6	13	0.157764
DEBR(8)	256	509	2	3.98	4	3	8	0.304482
FFT(6)	448	768	2	3.43	4	4	12	0.116233
SE(8)	256	381	1	2.98	3	4	15	0.152241
RAMAN-Q(3,17)	306	606	3	3.96	4	5	7	0.697224
RAMAN-C(3,7)	336	672	4	4.00	4	8	6	1.000000
HYP(8)	256	1024	8	8.00	8	4	8	2.000000

both cases we adapt the vertex degree in non static networks in every iteration to improve the convergence as shown in [9]. The additional simulation leads to an approximation of the fully balanced load amount. If its error is small enough, the second phase of the algorithm is entered. Each node now calculates its desired load based on its approximated and its current real integer load situation as it is described in Section 3. Then, the excess load units and the virtual negative tokens start performing a random walk until the error between the desired load and the real load becomes small enough.

4.1 Network topologies

The network types included in our tests are general interconnection topologies like Grid, Torus, FFT, Butterfly (BFLY), Shuffle-Exchange (SE), Cube Connected Cycle (CCC), de Bruijn (DEBR), and Hypercube (HYP) networks. All of these topologies are well known and some of them are designed to have many favorable properties for distributed computing, e.g. a small vertex degree and diameter and large connectivity. Furthermore, we also include Ramanujan (RAMAN) graphs that are described in [20]. An overview on some of the graphs' properties are displayed in Table 1. In all settings, $|V|^2$ tokens are initially placed on a single node and an edge fails with 10% probability. Furthermore, the nodes processing speed varies from 0.8 to 1.2.

Figure 1 demonstrates the algorithm's behavior on the 16×16 Torus. Applying FOS, the maximal weighted load in the system (left) is initially reduced quickly, while in the end many iterations for only small improvements are needed. At some point, due to the rounding, no further reduction will be possible. This is also visible when looking at the corresponding l_2 error (right, fos(int):error) which stagnates at around 200.0. In contrast, the error of the arbitrarily divisible load simulation (fos(real):error) converges steadily. Hence, the proposed load balancing algorithm switches to random walk in iteration 277. One can see (left) that this accelerates the load distribution and leads also to

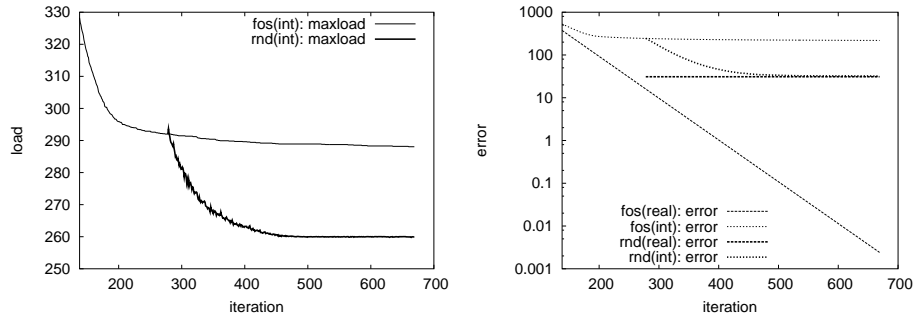


Figure 1: Balancing tokens on the 16x16 Torus.

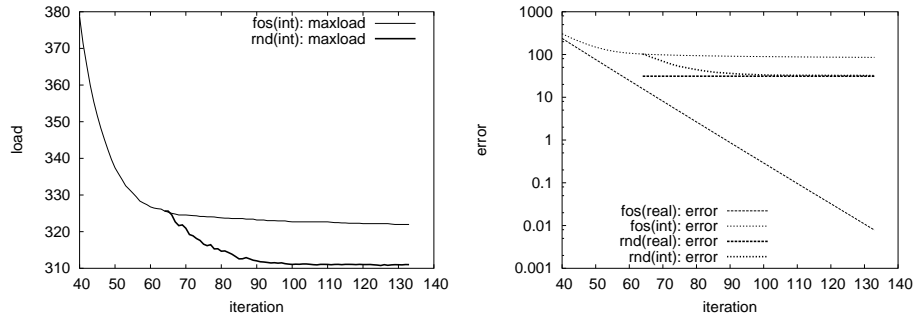


Figure 2: Balancing tokens on the Ramanujan Graph RAMAN-Q(3, 17).

a smaller maximal weighted load than FOS can ever achieve. Note, that the optimal load for each node is not exactly known when switching. Hence, the error of the unit-size token distribution (right, rnd(int):error) cannot exceed the one of the approximated solution (rnd(real):error).

A similar behaviour can be observed on the Ramanujan graphs and the Hypercube given in Figures 2 and 3. Compared to the Torus, these topologies possess much better properties (the second smallest eigenvalue of their Laplacian λ_2 is large), and therefore the diffusion process converges much faster. Note that although the Hypercube has better properties than the Ramanujan graph, its large degree and the resulting rounding operations prevent the distribution of indivisible load if using diffusion only. Hence, the benefit of the randomized approach is higher.

Further observations concerning varying parameters reveal that an increased edge failure probability slows the algorithm down. However, it also slightly improves the balance that can be achieved with diffusion, because the average vertex degree is smaller and therefore the rounding error decreases. Furthermore, we observe that the number of additional iterations needed is much smaller than expected regarding the theoretical increase of the bound through the additional logarithmic factor.

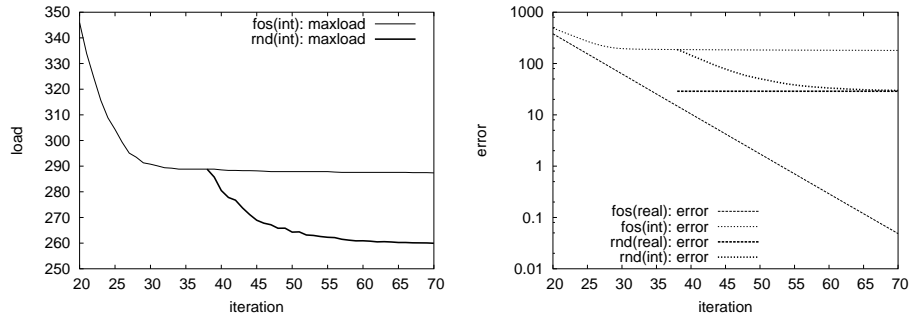


Figure 3: Balancing tokens on the Hypercube HYP(8).

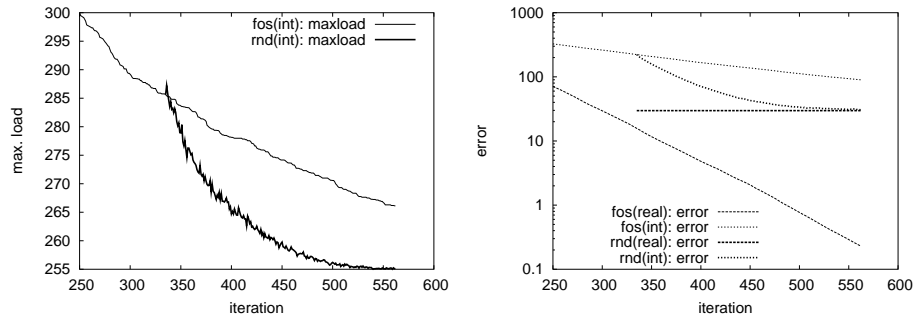


Figure 4: Balancing tokens in a mobile ad-hoc network with slow node movement.

4.2 Mobile ad-hoc networks

The second environment we use to simulate load balancing is a mobile ad-hoc network (Manet) model. The simulation area is the unit-square and 256 nodes are placed randomly within it. Prior to an iteration step of the general diffusion scheme, edges are created between nodes depending on their distance. Here, we apply the disc graph model (e.g. [28]) which simply uses a uniform communication radius for all nodes. After executing one load balancing step, all nodes move toward their randomly chosen way point. Once they have reached it, they pause for some iterations before continuing to their next randomly chosen destination. This model has been proposed in [19] and is widely applied in the ad-hoc network community to simulate movement. Note, that when determining neighbors as well as during the movement, the unit-square is considered to have wrap-around borders, meaning that nodes leaving on one side of the square will reappear at the proper position on the other side. For the experiments, we average the results from 25 independent runs.

The outcome of two experiment is shown in Figures 4 and 5. In both examples the communication radius is set to 0.1. While the nodes move quite slowly in the first setting with speeds between 0.001 and 0.005, they are ten times

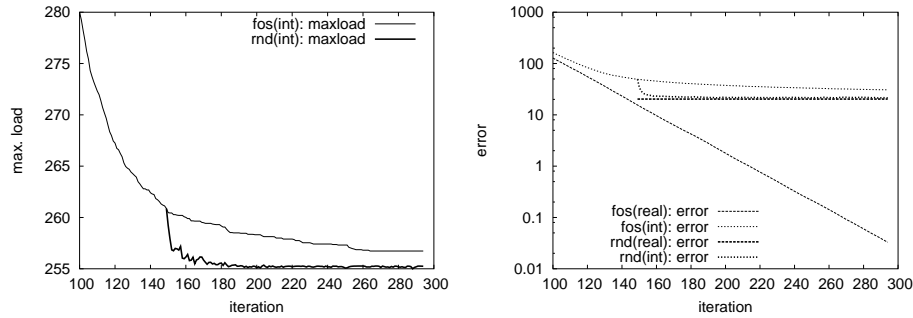


Figure 5: Balancing tokens in a mobile ad-hoc network with fast node movement.

faster in the second one. Furthermore, a node pauses for 3 iterations after it has reached its destination.

From the results one can conclude that, in contrast to the former results obtained for static network topologies, load can be better balanced when applying FOS only. This is due to the vertex movement which is an additional, and in later iterations the only mean to spread the tokens in the system. Higher movement rates will increase this effect. Nevertheless, the randomized algorithm is again able to speed up the load balancing process in both cases and is reaching an almost equalized state much earlier.

5 Conclusion

The presented randomized algorithm balances unit-size tokens in general heterogeneous and dynamic networks without global knowledge. It is able to reduce the weighted maximal overload in the system to $\mathcal{O}(1)$ with only slightly increasing of the run-time by at most a factor of $\mathcal{O}(\ln(n))$ compared to the general diffusion scheme. From our experiments, we can see that this additional factor is usually not needed in practice.

References

- [1] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.
- [2] F. R. K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional conference series in mathematics*. American Mathematical Society, 1997.
- [3] A. Cortes, A. Ripoll, F. Cedo, M. A. Senar, and E. Luque. An asynchronous and iterative load balancing algorithm for discrete load model. *Journal of Parallel and Distributed Computing*, 62:1729–1746, 2002.
- [4] D. M. Cvetkovic, M. Doob, and H. Sachs. *Spectra of Graphs*. Johann Ambrosius Barth, 3rd edition, 1995.
- [5] G. Cybenko. Load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.
- [6] R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25(7):789–812, 1999.
- [7] R. Elsässer and B. Monien. Load balancing of unit size tokens and expansion properties of graphs. In *Proceedings of SPAA '03*, pages 266–273, 2003.
- [8] R. Elsässer, B. Monien, and R. Preis. Diffusion schemes for load balancing on heterogeneous networks. *Theory of Computing Systems*, 35:305–320, 2002.
- [9] R. Elsässer, B. Monien, and S. Schamberger. Load balancing in dynamic networks. In *Proceedings of I-SPAN'04*, pages 193–200, 2004.
- [10] R. Elsässer, B. Monien, and S. Schamberger. Load balancing of indivisible unit size tokens in dynamic and heterogeneous networks. In *Proceedings of ESA '04*, pages 640–651, 2004.
- [11] O. Eriksen. A termination detection protocol and its formal verification. *Journal of Parallel and Distributed Computing*, 5:82–91, 1988.
- [12] J. A. Fill. Eigenvalue bounds on convergence to stationarity for nonreversible markov chains, with an application to the exclusion process. *The Annals of Applied Probability*, 1:62–87, 1991.
- [13] J. E. Gehrke, C. G. Plaxton, and R. Rajaraman. Rapid convergence of a local load balancing algorithm for asynchronous rings. *Theoretical Computer Science*, 220:247–265, 1999.
- [14] B. Ghosh, F. T. Leighton, B. M. Maggs, S. Muthukrishnan, C. G. Plaxton, R. Rajaraman, A. Richa, R. E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM Journal on Computing*, 29:29–64, 1999.

- [15] G. Golub. Bounds for the round-off errors in the richardson second order method. *BIT*, 2:212–223, 1962.
- [16] T. Hagerup and C. Rb. A guided tour of chernoff bounds. *Information Processing Letters*, 36(6):305–308, 1990.
- [17] M. E. Houle, A. Symvonis, and D. R. Wood. Dimension-exchange algorithms for load balancing on trees. In *Proceedings of Sirocco'02*, pages 181–196, 2002.
- [18] M. E. Houle, E. Tempero, and G. Turner. Optimal dimension-exchange token distribution on complete binary trees. *Theoretical Computer Science*, 220:363–376, 1999.
- [19] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.
- [20] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
- [21] M. Mihail. Conductance and convergence of markov chains: a combinatorial treatment of expanders. In *Proceedings of FOCS'89*, pages 526–531, 1989.
- [22] S. Muthukrishnan, B. Ghosh, and M. H. Schultz. First and second order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31:331–354, 1998.
- [23] Y. Rabani, A. Sinclair, and R. Wanka. Local divergence of markov chains and the analysis of iterative load-balancing schemes. In *Proceedings of FOCS'98*, pages 694–703, 1998.
- [24] S. Rönn and H. Haikkonen. Distributed termination detection with counters. *Information Processing Letters*, 34:223–227, 1990.
- [25] P. Sanders. Analysis of nearest neighbor load balancing algorithms for random loads. *Parallel Computing*, 25:1013–1033, 1999.
- [26] V. S. Sunderarm and G. A. Geist. Heterogeneous parallel and distributed computing. *Parallel Computing*, 25:1699–1721, 1999.
- [27] B. Szymanski, Y. Shi, and S. Prywes. Synchronized distributed termination. *IEEE Transactions on Software Engineering*, 11:1136–1140, 1985.
- [28] Y. Wang and X.-Y. Li. Geometric spanners for wireless ad hoc networks. In *Proceedings of ICDCS'02*, pages 171–180. IEEE, 2002.