Diffusion Schemes for Load Balancing on Heterogeneous Networks*

Robert Elsässer, Burkhard Monien and Robert Preis
Department of Mathematics and Computer Science
University of Paderborn, Germany
{elsa,bm,robsy}@uni-paderborn.de

Abstract

Several different diffusion schemes have previously been developed for load balancing on homogeneous processor networks. We generalize existing schemes, in order to deal with heterogeneous networks. The generalized schemes may operate efficiently on networks where every processor can have arbitrary computing power, i. e., the load will be balanced proportionally to these weights. The balancing flow that is calculated by the schemes for homogeneous networks is minimal with regard to the l_2-norm and we prove this to hold true for the generalized schemes, too. We demonstrate the usability of the generalized schemes by a number of experiments on several heterogeneous networks.

1 Introduction

Load balancing is an essential task for the efficient use of parallel computer systems. It can be observed that the work loads of many parallel applications have a dynamic behavior and can change dramatically during runtime. To achieve an efficient use of the processor network, the load has to be balanced between the processors during runtime. Obviously, the balancing scheme itself is supposed to be highly efficient in order to ensure an overall benefit of the parallel computation.

One major field of application is parallel adaptive finite element simulations where a geometric space, discretized by using a mesh, is partitioned into sub-regions. The computation proceeds on mesh elements in each sub-region independently. As the computation proceeds the mesh refines and coarsens, depending on the kind of problem such as turbulence or shocks (in the case of fluid dynamics simulations, for example). Furthermore, the size of the sub-regions (in terms of the numbers of elements) has to be balanced. In a dynamic behavior of the underlying application, such modifications of the mesh may occur very often, and there is need to balance the load among processors for several times during the parallel computation. Each time when the application is stopped to balance the load, a balancing problem on a static load distribution has to be executed. Although the application behaves dynamically, we can concentrate on the solving of a static load balancing problem. The problem of parallel finite element simulations has been extensively studied - see the book [FWM94] for an excellent selection of applications, case studies, and references.

*Supported by the German Research Foundation (Deutsche Forschungsgemeinschaft) Project SFB-376 and by the European Union program IST-1999-14186 (ALCOM-FT).

In order to apply a load balancing algorithm, it must be possible to divide the work load of a processor and to migrate load items during runtime. A common approach is to solve the load balancing problem in two steps. Firstly, a balancing flow of the amount of load that has to be migrated is calculated. Secondly, the load items are selected and migrated according to that flow. In this paper we deal with step one, i. e., we will explore the question of calculating the amount of flow of the migrated load so that, after the load was migrated, a balanced processing time among all processors is achieved.

An important aspect in this context is the local calculation of the flow. The calculation has to be very fast, in order not to increase the overall computation time of the application too much. Thus, a global gathering of the load information and the distribution of the calculated flow over the whole network results in a high communication overhead. Therefore, we focus on local iterative load balancing schemes in which each processor exchanges information only with its neighbors. Consequently, after a small number of iterations a globally balanced status is achieved.

So far, only little work has been done to address load balancing algorithms for heterogeneous networks. Heterogeneous networks consist of processors with different computing power or different memory capacity, or they have different communication links. Consult [SG99] for an overview of the evolution of heterogeneous concurrent computing in the context of the parallel virtual machine (PVM) system. Heterogeneous networks are extremely attractive because they often appear in computer networks which consist of processors from different manufacturers and of different types.

In [DFM99] the authors also consider networks with different communication speeds between the processors. They show that the existing balancing schemes can be generalized easily such that, roughly speaking, fast communication links get a higher load migration volume than others. This generalization of schemes for different communication speeds can be combined with our new generalization of the computational power of the processors. Therefore, it is now possible to address the two major kinds of heterogeneous networks together.

We consider the following load balancing problem in heterogeneous networks. The processors of the network have different work loads and different processor speeds which can result in unequal remaining processing times. The load has to be balanced among the processors proportionally to their computing power. More formally, given a network with n nodes where each node i contains a work load of w_i and a weight of c_i, calculate a load balancing flow over the edges of the network such that, after termination, each node i has the balanced proportional work load of

$$\overline{w_i} := \frac{\sum_{j=1}^{n} w_j}{\sum_{j=1}^{n} c_j} c_i \ .$$

The result is that the time to consume the load on each node is approximately equal. Obviously, this problem generalizes the load balancing on homogeneous networks where $c_i = 1$ for all $1 \leq i \leq n$. We assume that the situation is fixed, i. e., no load is generated or consumed during the balancing process and the structure of the network is not changed.

If the global imbalance vector $w - \overline{w}$ is known, it is possible to solve the problem by solving a linear system of equations. This has been done for homogeneous networks in [HBE98]. If only local information about the load imbalance is to be used, the nodes of the graph have to balance the load with their neighbors iteratively until the whole network is globally balanced. The class of local iterative load balancing algorithms distinguishes between *diffusion* [Boi90, Cyb89, DFM99, DMN97, GMS98, HBE98, XL97] and *dimension* *exchange* [Cyb89, XL97] methods. The two types differ in their way of communicating. Diffusion algorithms assume that a node of the graph can send and receive messages to/from all its neighbors simultaneously in any iteration, whereas dimension exchange algorithms assume that in each iteration a node of the graph can send and receive messages to/from only one of its neighbors, i. e., only pairwise communication is possible within an iteration. The diffusion matrix in the First-Order scheme (FOS, Section 3) has the same structure as the transition matrix of a reversible ergodic Markov chain. Besides, a balanced situation can be regarded as the stationary distribution of this Markov chain.

The *alternating-direction-iterative* scheme [EFMP99] is a mixture of the diffusion and the dimension exchange methods. It reduces the number of iteration steps for networks which can be represented as cartesian products of graphs. However, the resulting flow of this scheme might have load-migration loops of infinitely large values. This would lead to a high load-migration overhead. In [DMP00], the *multi-diffusion* scheme for cartesian products of graphs is proposed. It ensures that the resulting flow cannot contain any load-migration loops.

The *quality* of a balancing algorithm is judged by two criteria. Firstly, the numbers of steps that are required to achieve a balanced state. In Section 3 we will discuss the rate of convergence for several schemes. Secondly, the amount of load moved over the edges of the graph. In [DFM99] it is shown that all of the diffusion schemes calculate the same flow. Furthermore, they showed that this flow is minimal with regard to the l_2-norm. It ensures that there are no loops in the flow of the load. Generally, the l_2-minimal flow has also small values in the l_1- and l_∞-norm, leading to a small load-migration time in parallel computation.

The amount of load migration might be much higher in heterogeneous than in homogeneous networks, depending on the heterogeneous structure of the network. Nevertheless, in this paper we show that the resulting balancing flow of the generalized schemes is also minimal in the l_2-norm. Thus, the schemes discussed in this paper are highly suitable for load balancing on heterogeneous networks.

We will see in Section 3 that the rate of convergence for the balancing schemes is closely coupled to the second smallest eigenvalue of a generalized Laplacian which is defined in the following section. Therefore, it is important to gain more knowledge about that eigenvalue and to have a good approximation to it. We discuss a relation between this eigenvalue and the expansion of the heterogeneous network. For homogeneous networks, the relation between the second eigenvalue of the network and its expansion has been studied in [DK86, Moh89]. Analogue bounds have been developed for the spectral gap based on the conductance of a reversible Markov chain in [DS91, SJ89].

The main contributions of this paper are the following.

- We derive a generalization of the known diffusive load balancing schemes (i) First-Order FOS, (ii) Second-Order SOS, (iii) Chebyshev CHEBY and (iv) Optimal OPT in order to balance the load proportional to the weights of the processors (Section 3.1).
- We show that all generalized schemes compute the unique flow which is minimal in the l_2-norm (Section 3.2).
- We perform several experiments on different weight scenarios on some exemplary networks and show that the schemes for homogeneous networks can be used with small modifications to work on heterogeneous networks (Section 4).

A preliminary version of this paper appeared in the proceedings of the Symposium on Parallel Algorithms and Architectures (SPAA 2000) [EMP00].

2 Definitions and Background

Let $G = (V, E)$ be a connected, undirected graph with $|V| = n$ nodes and $|E| = N$ edges. Node $v_i \in V$ has a *weight* of $c_i \in I\!R$ and a *load* of $w_i \in I\!R$. Let $w \in I\!R^n$ be the vector of load values. Our goal is to balance the load proportionally to the weight. Therefore,

$$\overline{w} := \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} c_i}(c_1, \ldots, c_n)$$

denotes the vector of a proportionally balanced load.

Denote the $n \times n$ diagonal matrix of the weights with C, where c_i are the diagonal entries and 0 elsewhere. Define $A \in \{-1, 0, 1\}^{n \times N}$ to be the node-edge *incidence matrix* of G. A contains a row for every node and a column for every edge. Every column has exactly two non-zero entries "1" and "-1" for the two nodes incident to the corresponding edge. The signs of these non-zeros (implicitly) define directions for the edges of G. These directions will be used later on to express the direction of the flow. Let $B \in \{0, 1\}^{n \times n}$ be the *adjacency matrix* of G. Since G is undirected, B is symmetric. Column/row i of B contains 1's at the positions of all neighbors of v_i. For our construction we need the *Laplacian* $L \in Z\!\!Z^{n \times n}$ of G defined as $L := D - B$, with $D \in I\!N^{n \times n}$ containing the node degrees as diagonal entries and 0 elsewhere. It is easy to verify that $L = AA^T$.

The Laplacian L and its eigenvalues are used for the load balancing on homogeneous networks. For heterogeneous networks we have to use the generalized Laplacian LC^{-1}.

Let $x \in I\!R^N$ be a flow on the edges of G. Let x be called a *balancing flow* on G iff

$$Ax = w - \overline{w}.$$

It shows that at each node the flow balance is the difference between its initial load and its proportional load value, i. e., after migrating a load of x_e via edge $e \in E$, the load is globally balanced.

We will consider *local iterative balancing algorithms* and we first illustrate the first order scheme (FOS) by Cybenko [Cyb89]. In each iteration of this method, each node $v_i \in V$ performs local communication with its neighbors only.

$$\begin{aligned} \forall e &= \{v_i, v_j\} \in E \text{ perform} \\ y_e^{k-1} &= \alpha\left(\frac{w_i^{k-1}}{c_i} - \frac{w_j^{k-1}}{c_j}\right); \quad x_e^k = x_e^{k-1} + y_e^{k-1}; \end{aligned} \tag{1}$$

and

$$w_i^k = w_i^{k-1} - \sum_{e=\{v_i, v_j\} \in E} y_e^{k-1}. \tag{2}$$

Here, y_e^k is the amount of load sent via edge e in step k. x_e^k is the total load sent via edge e until iteration k and w_i^k is the load of the node i after the k^{th} iteration. Parameter α controls the ratio between load difference and the load that has to be moved. It determines the rate of convergence. For an appropriate choice of α, iteration (2) converges to the proportionally balanced load $\overline{w}$. The iteration can be written in matrix notation as $w^k = Mw^{k-1}$ with the *diffusion matrix* $M = I - \alpha LC^{-1} \in I\!R^{n \times n}$. We will see in Section 3.2 that it results in an l_2-minimal flow (the l_2 norm of x^k is defined as $\|x^k\| = \left(\sum_{i=1}^{N}(x_i^k)^2\right)^{1/2}$).

The generalizations of diffusive schemes for networks with different communication speeds have previously been discussed in [DFM99]. In this case the Laplacian is defined as $\tilde{L} = \tilde{A}\tilde{A}^T$, where $\tilde{A} = AF^{-1}$ and the matrix F denotes the $N \times N$ diagonal matrix with $F_{ii} = \sqrt{f_i}$ for all $1 \leq i \leq N$. With f_i we denote the edge-weight of the edge i. We denote with $0 = \lambda_1 < \ldots < \lambda_m$ the m distinct eigenvalues of $\tilde{L}C^{-1}$. Now the general form of the first order scheme presented in equation (1) is

$$\begin{aligned} \forall e &= \{v_i, v_j\} \in E \text{ perform} \\ y_e^{k-1} &= \frac{\alpha}{f_e}\left(\frac{w_i^{k-1}}{c_i} - \frac{w_j^{k-1}}{c_j}\right); \quad x_e^k = x_e^{k-1} + y_e^{k-1}; \end{aligned}$$

and

$$w_i^k = w_i^{k-1} - \sum_{e=\{v_i,v_j\}\in E} y_e^{k-1} \ .$$

In the following we consider the eigenvalues of the Laplacian of a node-weighted graph. In the case of homogeneous networks, the eigenvalues of the positive-semidefinite Laplacian L are used to determine an optimal value for α. However, for the heterogeneous networks we have to work with the generalized Laplacian $\tilde{L}C^{-1}$ and we analyze its eigenvalues in the following lemma.

Lemma 1 *The eigenvalues of the matrix $\tilde{L}C^{-1}$ are real and non-negative. Moreover, they form a basis in $\mathbb{R}^n$.*

Proof: First we prove that $\tilde{L}C^{-1}$ has the same eigenvalues as $C^{-1/2}\tilde{L}C^{-1/2}$. Since it is a real positive-semidefinite symmetric matrix, $\tilde{L}C^{-1}$ does only have non-negative real eigenvalues.

Let $C^{-1/2}\tilde{L}C^{-1/2}u = \lambda u$. Then it is $\tilde{L}C^{-1}C^{1/2}u = LC^{-1/2}u = C^{1/2}C^{-1/2}\tilde{L}C^{-1/2}u = \lambda C^{1/2}u$. It holds that $C^{1/2}u$ is an eigenvector of $\tilde{L}C^{-1}$ to the eigenvalue λ. For any $x \in \mathbb{R}$ it holds $(\frac{1}{\sqrt{c_1}}x_1, \ldots, \frac{1}{\sqrt{c_n}}x_n) = \sum_{i=1}^n \beta_i u_i$ for some constants β_i. It follows that $x = \sum_{i=1}^n \beta_i C^{1/2} u_i$. $\bowtie$

If α is chosen appropriately, the diffusion matrix $M = I - \alpha\tilde{L}C^{-1}$ has the distinct eigenvalues $1 = \mu_1 > \mu_2 > \ldots > \mu_m > -1$ and $(c_1, c_2, \ldots, c_n)$ is an eigenvector of M to the simple eigenvalue $\mu_1 = 1$. We define $\gamma = \max\{|\mu_2|, |\mu_m|\} = \max\{|1-\alpha\lambda_2|, |1-\alpha\lambda_m|\}$. A small γ will lead to a fast rate of convergence (Section 3.2).

The minimum of γ is achieved for $\alpha = \alpha_{opt} = \frac{2}{\lambda_2+\lambda_m}$ which is the same value as for homogeneous networks and the eigenvalues of L. If we use $\alpha = \alpha_{opt}$, it is easy to see that $\gamma = \frac{\lambda_m - \lambda_2}{\lambda_m + \lambda_2}$. It shows that a large second smallest eigenvalue λ_2 will lead to a small γ and, therefore, to a fast rate of convergence. Therefore, we will discuss a relation between λ_2 and the expansion of the network in Section 3.3.

The values of α_{opt} and γ can be calculated for heterogeneous networks in the same way as for homogeneous networks. The only difference is the use of the eigenvalues of $\tilde{L}C^{-1}$ instead of L.

Definition 1 *Any scheme for which the work load w^k in step k can be expressed in the form of*

$$w^k = p_k(M)w^0 \quad \textit{where } p_k \in \overline{\Pi}_k \tag{3}$$

is called a polynomial based *load balancing scheme.* $\overline{\Pi}_k$ *denotes the set of all polynomials* p *of degree* $\deg(p) \le k$ *satisfying the constraint* $p(1) = 1$.

Note that the sum over the entries in each column in M is equal to 1. Furthermore, the condition $p_k(1) = 1$ implies that all column sums in the matrix $p_k(M)$ are 1. Thus, the total work load is conserved, i. e., $\sum_{i=1}^n w_i^k = \sum_{i=1}^n w_i^0$. The representation (3) is primarily useful for the mathematical analysis. Indeed, for (3) defining an algorithmically feasible nearest neighbor scheme, it must be possible to rewrite it as an update process where w^k is computed from w^{k-1} (and maybe from some more previous iterates) involving *one* multiplication with M only. This means that the polynomials p_k have to satisfy some kind of short recurrence relation. Such relations will be explicitly stated in some special cases that will be discussed in the following.

The convergence of a polynomial based scheme will depend on whether (and how fast) the 'error' $e^k = w^k - \overline{w}$ between the iterate $w^k = p_k(M)w^0$ and the balanced load $\overline{w} = \frac{\sum_{i=1}^n w_i^0}{\sum c_i}(c_1, \ldots, c_n)$ will tend to zero. These errors e^k have two fundamental properties which are stated in the next lemma.

Lemma 2 *Let* $M = I - \alpha\tilde{L}C^{-1}$, *let* z_i *be the eigenvectors of* $\tilde{L}C^{-1}$ *and* $w^0 = \sum_{i=1}^m \beta_i z_i$ *for some constants* β_i. *Then*

$$e^0 = \sum_{i=2}^m \beta_i z_i, \tag{4}$$

$$e^k = p_k(M)e^0, \quad k = 0, 1, 2, \ldots \tag{5}$$

Proof: Obviously, $\overline{w} = \beta_1 z_1$. The first equality follows from $w^0 = e^0 + \overline{w}$. To show (5), we note that due to $p_k(1) = 1$ the vector $\overline{w}$ is an eigenvector of $p_k(M)$ with eigenvalue 1. Thus,

$$e^k = w^k - \overline{w} = p_k(M)(w^0 - \overline{w}) = p_k(M)e^0 .$$

$\bowtie$

3 Loadbalancing Schemes

In this section we first generalize the load balancing schemes FOS, SOS, CHEBY and OPT for heterogeneous networks and determine their convergence. Then, we show that all generalized schemes compute the unique flow which is minimal in the l_2-norm. Finally, we discuss the relation between the second smallest eigenvalue and the expansion (see section 3.3 for the definition) of the network.

3.1 Polynomial Iterative Methods

By the use of both statements from Lemma 2 we see that the error e^k of any polynomial based scheme satisfies

$$e^k = p_k(M)\left(\sum_{i=2}^m \beta_i z_i\right) = \sum_{i=2}^m p_k(M)\beta_i z_i$$

$$= \sum_{i=2}^{m} p_k(\mu_i)\beta_i z_i \ .$$

Here we made use of $p_k(M)\beta_i z_i = p_k(\mu_i)\beta_i z_i$, since z_i is an eigenvector of M with eigenvalue μ_i (z_i is also an eigenvector of $\tilde{L}C^{-1}$, but to a different eigenvalue). This fundamental relation allows us to analyze several nearest neighbor load balancing schemes in detail. With u_i, $i = 1, \dots, n$, we denote the eigenvectors of the matrix $C^{-1/2}\tilde{L}C^{-1/2}$. We know that they form a basis and that they are orthogonal to each other and

$$z_i = C^{1/2}u_i.$$

We obtain

$$\begin{aligned}
[e^k] &= \sum_{i=2}^{m} p_k(\mu_i)^2[\beta_i z_i] \\
&\leq \left([C^{1/2}] \max_{i=2}^{m} p_k(\mu_i)\right)^2 \cdot \sum_{i=2}^{m} [\beta_i u_i] \\
&\leq \left([C^{1/2}][C^{-1/2}]p_k(\gamma)\right)^2 [e^0]
\end{aligned}$$

where we denote with $[\ ]$ the value $\|\ \|_2^2$.

It holds that $\gamma = \max_{i=2}^{m} |\mu_i|$ and by the use of $\alpha_{opt} = \frac{2}{\lambda_2+\lambda_m}$ we obtain $\gamma = \frac{\lambda_m-\lambda_2}{\lambda_m+\lambda_2}$. We can observe that λ_2 plays an important role in the convergence of the local iterative schemes. Therefore, in Section 3.3 we discuss bounds for λ_2.

First-Order-Scheme (FOS)

We start our analysis of different methods with the FOS scheme of Cybenko [Cyb89], where one takes $p_k(t) = t^k$. These polynomials satisfy the simple short recurrence $p_k(t) = t \cdot p_{k-1}(t), k = 1, 2, \dots$, so that we have, with $M = I - \alpha\tilde{L}C^{-1}$,

$$w^k = Mw^{k-1},\ k = 1, 2, \dots\ .$$

In this situation $|p_k(\mu_i)| = |\mu_i^k| \leq \gamma^k$ for $i = 2, \dots, m$, where $\gamma = \max_{i=2}^{m} |\mu_i|$. Thus, we obtain

$$\|e^k\|_2 \ \leq \ \sqrt{\frac{c_{max}}{c_{min}}}\gamma^k \cdot \|e^0\|_2 \ .$$

An analogue consideration has been made for reversible ergodic Markov chains, where the convergence of the Markov chain to a stationary distribution has been developed [DS91, SJ89]. The matrix M can be viewed as a transition matrix with the transition probabilities $1/c_j$ from state i to state j and the stationary distribution corresponds to the balanced situation $(c_1, \dots, c_n)$. The Markov chain is ergodic, because $(M_{ij})c_j = (M_{ji})c_i$ for all $0 \leq i, j \leq n-1$.

Second-Order-Scheme (SOS)

The SOS of [GMS98] takes the polynomials

$$\begin{aligned}
p_0 &\equiv 1,\ p_1(t) = t\ , \\
p_k(t) &= \beta t p_{k-1}(t) + (1-\beta)p_{k-2}(t)\ ,\ \ k = 2, 3, \dots
\end{aligned}$$

so that

$$\left.\begin{array}{rcl} w^1 & = & Mw^0 \ , \\ w^k & = & \beta M w^{k-1} + (1-\beta) w^{k-2} \ , \ \ k = 2,3,\dots \end{array}\right\} \tag{6}$$

Here, β is a fixed parameter. This scheme is known as the second order Richardson method in numerical analysis. Investigations in [GV61] show that this iteration converges to $\overline{w}$ whenever $\beta \in (0,2)$ and that the fastest convergence occurs for

$$\beta = \beta_{opt} = 2/\left(1 + \sqrt{1-\gamma^2}\right) . \tag{7}$$

In this case, we have (see [GV61, Var62])

$$\max_{t\in[-\gamma,\gamma]} |p_k(t)| \ = \ (\beta_{opt} - 1)^{\frac{k}{2}} \left(1 + k\sqrt{1-\gamma^2}\right) \ .$$

Since $\max_{i=2}^m |p_k(\mu_i)| \le \max_{t\in[-\gamma,\gamma]} |p_k(t)|$ we get

$$\|e^k\|_2 \ \le \ \sqrt{\frac{c_{max}}{c_{min}}} (\beta_{opt} - 1)^{\frac{k}{2}} \left(1 + k\sqrt{1-\gamma^2}\right) \cdot \|e^0\|_2 \ . \tag{8}$$

According to [GMS98], by comparing the factors γ^k of FOS and $(\beta_{opt}-1)^{\frac{k}{2}}(1+k\sqrt{1-\gamma^2})$ of SOS (for γ close to 1), this can be interpreted as SOS being of 'second' order whereas FOS only of 'first' order.

Chebyshev-Scheme (CHEBY)

The Chebyshev method differs from SOS only in the fact that parameter β does now depend on k according to

$$\beta_1 \ = \ 1, \quad \beta_2 = \frac{2}{2-\gamma^2}, \ \text{and} \quad \beta_k = \frac{4}{4-\gamma^2\beta_{k-1}}, \tag{9}$$

for $k = 3,4,\dots$. The corresponding polynomials p_k are the (scaled) Chebyshev polynomials for the interval $[-\gamma,\gamma]$. This means that they are optimal in the sense that (see [GV61, Var62])

$$\begin{array}{rcl} \max\limits_{t\in[-\gamma,\gamma]} |p_k(t)| & = & \min\limits_{p\in\overline{\Pi}_k} \max\limits_{t\in[-\gamma,\gamma]} |p(t)| \\ & = & (\beta_{opt}-1)^{\frac{k}{2}} \dfrac{2}{1+(\beta_{opt}-1)^k}. \end{array}$$

Similarly to SOS, this yields the estimate

$$\|e^k\|_2 \ \le \ \sqrt{\frac{c_{max}}{c_{min}}} (\beta_{opt}-1)^{\frac{k}{2}} \frac{2}{1+(\beta_{opt}-1)^k} \cdot \|e^0\|_2 \ . \tag{10}$$

The factor in (10) is always smaller than in (8) which shows that the Chebyshev method is usually preferred over the SOS scheme. Asymptotically, however, both methods can be regarded to perform identically since

$$\begin{array}{l} \lim\limits_{k\to\infty} \left[(\beta_{opt}-1)^{\frac{k}{2}} \dfrac{2}{1+(\beta_{opt}-1)^2}\right]^{\frac{1}{k}} \quad = \\ \lim\limits_{k\to\infty} \left[(\beta_{opt}-1)^{\frac{k}{2}} \left(1+k\sqrt{1-\gamma^2}\right)\right]^{\frac{1}{k}} \quad = \quad (\beta_{opt}-1)^{\frac{1}{2}} \ . \end{array}$$

Optimal-Scheme (OPT)

The knowledge of all distinct eigenvalues of $\tilde{L}C^{-1}$ of the network is needed for OPT [DFM99]. However, it guarantees a completely balanced work load after $m-1$ iterations. Because $m \leq n$, the number of iterations does never exceeds the number of nodes in the network. We generalize a very simple optimal scheme [EFMP99] in the following and apply the local iterative algorithm with the distinct non-zero eigenvalues λ_k of $\tilde{L}C^{-1}$, $2 \leq k \leq m$, (in any order) as

$$\begin{aligned} &\forall\, e = \{v_i, v_j\} \in E : \\ &\qquad y_e^{k-1} = \frac{1}{\lambda_k f_e}(w_i^{k-1}/c_i - w_j^{k-1}/c_j); \\ &\qquad x_e^k = x_e^{k-1} + y_e^{k-1}; \\ &\text{and} \quad w_i^k = w_i^{k-1} - \sum_{e=\{v_i,v_j\}\in E} y_e^{k-1} \, . \end{aligned}$$

Each node i simply adds a flow of $\frac{1}{\lambda_k f_e}(w_i^{k-1} - w_j^{k-1})$ to the flow over edge $e = \{i, j\}$ in iteration k, choosing a different eigenvalue for each iteration. Therefore,

$$w^k = \left(I - \frac{1}{\lambda_{k+1}}\tilde{L}C^{-1}\right) w^{k-1} \, .$$

For error e^{m-1} after $m-1$ iterations we obtain

$$e^{m-1} = \prod_{i=2}^{m} \left(I - \frac{1}{\lambda_i}\tilde{L}C^{-1}\right) \sum_{j=2}^{m} z_j = 0.$$

The order of the eigenvalues can be arbitrarily. However, in [EFMP99] it is shown that one may trap in numerically instable conditions and it is shown how to avoid them.

A similar optimal scheme based on the same background was developed for homogeneous networks in [HB99].

When comparing the different schemes, SOS is significantly faster than FOS and CHEBY is asymptotically not faster than SOS. All these schemes usually need much more than $m-1$ iterations, whereas OPT does never need more than $m-1$ iterations. Although the calculation of all eigenvalues for OPT can be very time consuming, for a given network we have to calculate them only once.

3.2 Solution Quality

In this section we consider the quality of the flow computed by the polynomial schemes presented in the previous subsection. The results of [DFM99] obtained for homogeneous networks can also be generalized for heterogeneous networks. In the following we consider the next two lemmas from [DFM99] which will be used to prove that the flow is minimal according to the l_2-norm.

Lemma 3 ([DFM99]) *The equation $\tilde{L}z = b$ has a solution (and then infinitely many), if and only if $b \in (1, 1, \ldots, 1)^{\perp}$.*

We can now state the characterization of l_2-minimal flows.

Lemma 4 ([DFM99]) *Consider the l_2 minimization problem*

$$\textit{minimize } \|x\|_2 \textit{ over all } x \textit{ with } \tilde{A}x = b.$$

Provided that $b \in (1, 1, \dots, 1)^\perp$, the solution to this problem is given by

$$x = \tilde{A^T} z, \textit{ where } \tilde{L}z = b. \tag{11}$$

If we solve the equation $\tilde{L}z = \overline{w} - w^0$ and set $x = \tilde{A^T}z$, then we obtain the l_2-minimal flow for heterogeneous networks. $\tilde{L}z = \overline{w} - w^0$ has a solution, because, as it is easy to see, $\overline{w} - w^0$ is perpendicular to $(1, \dots, 1)$.

In the following lemma we consider iterative methods where we have a sequence of work loads converging to the average load. For this algorithm the difference vector $\overline{w} - w^0$ does not have to be computed in advance.

Our next lemma shows that if we have a sequence of work loads converging to the average load, and if we have an l_2-minimal flow for each such load, then these minimal flows converge to the minimal flow for the average load.

Lemma 5 *Let w^k be a (finite or infinite) sequence of work loads which converges to the average load $\overline{w}$. Moreover, let*

$$w^k = w^0 + \tilde{A}x^k$$

be such that $\|x^k\|_2$ is minimal, i. e., (by Lemma 4)

$$x^k = \tilde{A^T} z^k, \textit{ where } \tilde{L}z^k = w^k - w^0 \ .$$

Then there is $\lim_{k\to\infty} x^k = \bar{x}$, $\overline{w} = w^0 + \tilde{A}\bar{x}$, and $\|\bar{x}\|_2$ is minimal.

Proof: Lemma 5 holds for every $\overline{w}$ such that $\overline{w} - w^0$ is perpendicular to $(1, \dots, 1)$. In [DFM99] the lemma was only proven for homogeneous networks where $\overline{w} = \beta_1(1, \dots, 1)$. For heterogeneous networks it also holds that $\overline{w} - w^0$ is perpendicular to $(1, \dots, 1)$ and therefore the same proof works in this case, too. $\bowtie$

Now we are ready to state the main theorem, which shows that the polynomial schemes stated in the previous section provide an l_2-minimal flow.

Theorem 1 *Let $p \in \overline{\Pi}_k$ be a polynomial that satisfies the 3-term recurrence equation*

$$p_k(t) = (\sigma_k t - \tau_k)p_{k-1}(t) + \rho_k p_{k-2}(t) \ , \tag{12}$$

with

$$\rho_1 = 0 \textit{ and } \sigma_k - \tau_k + \rho_k = 1 \quad \forall \ k = 1, 2, \dots. \tag{13}$$

Let

$$d^0 = -\alpha\sigma_1 C^{-1} w^0 \, ; \quad x^1 = y^0 = \tilde{A^T} d^0 \, ; \quad w^1 = w^0 + \tilde{A}y^0 \, ;$$

and for $k = 2, 3, \dots$

$$\left.\begin{array}{lcl} d^{k-1} & = & -\alpha\sigma_k C^{-1} w^{k-1} - \rho_k d^{k-2} \, ; \\ y^{k-1} & = & \tilde{A^T} d^{k-1} \, ; \\ x^k & = & x^{k-1} + y^{k-1} \, ; \\ w^k & = & w^{k-1} + \tilde{A}y^{k-1} \, ; \end{array}\right\} \tag{14}$$

be the update process for x^k and w^k.

Then, $w^k = p_k(I-\alpha\tilde{L}C-1)w^0$, $\lim_{k\to\infty} x^k = \bar{x}$ *and there is* $\lim_{k\to\infty} w^k = \overline{w}$, $\overline{w} = w^0 + \tilde{A}\bar{x}$ *and* $\|x\|_2$ *is minimal.*

Proof: Using the matrix notation we obtain for $k = 1$ that $w^1 = w^0 + \tilde{L}d^0 = p_1(I - \alpha\tilde{L}C^{-1})w^0$. We assume that for any $i < k$ it holds $w^i = p_i(I - \alpha\tilde{L}C^{-1})w^0$. We get

$$\begin{aligned} w^k &= w^{k-1} + \tilde{A}y^{k-1} = w^{k-1} + \tilde{L}d^{k-1} \\ &= w^{k-1} - \alpha\sigma_k\tilde{L}C^{-1}w^{k-1} - \rho_k\tilde{L}d^{k-2} \\ &= w^{k-1} - \alpha\sigma_k\tilde{L}C^{-1}w^{k-1} - \rho_k(w^{k-1} - w^{k-2}) \\ &= p_k(I - \alpha\tilde{L}C^{-1})w^0 \end{aligned}$$

Using $z_k = \sum_{i=0}^{k-1} d^{k-1}$ and $w^k = w^{k-1} + \tilde{L}d^{k-1}$ we obtain $\tilde{L}z^k = w^k - w^0$. Furthermore, $x^k = x^{k-1} + \tilde{A}^T d^{k-1}$ and $x^1 = \tilde{A}^T d^0$ results in $x^k = A^T z^k$.

Therefore, Theorem 1 follows from Lemma 5. $\bowtie$

Looking at the schemes discussed so far, we have $\sigma_k = 1,\ \tau_k = \rho_k = 0$ for the FOS so that $d^{k-1} = -\alpha w^{k-1}$. We have $\sigma_k = \beta_k,\ \tau_k = 0,\ \rho_k = (1-\beta_k)$ in the Chebyshev scheme for each one but the first step. This yields $d^{k-1} = -\alpha\beta_k w^{k-1} - (1-\beta_k)d^{k-2}$. The first step is identical to FOS. The SOS scheme does only differ from Chebyshev by the fact that $\sigma_k = \beta_{opt}$.

3.3 Relation between the Second Smallest Eigenvalue and the Expansion

We have seen in the previous section that the second smallest eigenvalue of $\tilde{L}C^{-1}$ plays an important role for the convergence behavior of the balancing schemes. It is often used in order to optimize the balancing parameters.

The relation between the second smallest eigenvalue of the Laplacian and the expansion of a graph was discussed in a great detail [Che70, DK86, Moh89]. A similar relation between the Laplace operator of a reversible ergodic Markov chain and the conductance was developed by Sinclair in [Sin93]. Using analog techniques from the linear algebra we provide a relation for the second smallest eigenvalue of $\tilde{L}C^{-1}$ and the expansion of a weighted graph. We define

$$\iota_c(G) = min_{S\subset V(G)} \frac{E(S,\overline{S})}{min\{vol(S), vol(\overline{S})\}},$$

where $E(S,\overline{S})$ represents the weight of cut edges between S and $\overline{S}$ and $vol(S) = \sum_{i\in S} c_i$. Now we can state the following theorem:

Theorem 2 *For a vertex weighted graph it holds*

$$\frac{\iota_c^2(G)}{2} min_i(\frac{c_i}{deg_i}) \leq \lambda_2(\tilde{L}C^{-1}) \leq 2\iota_c(G)\ .$$

This result can be obtained from the relation between the second smallest eigenvalue of a transition matrix of an ergodic, reversible Markov chain and the conductance of this Markov chain [Sin93].

4 Experiments

In this section, we present the results of experiments with the loadbalancing schemes described in the previous section on heterogeneous networks. We focus on heterogeneous networks with different weights for the nodes.

We choose different networks. Each of them consists of 64 nodes. They are the path PATH64, the square grid GRID8x8 and the hypercube HYP6. We experiment with the following heterogeneous node weights.

HOMO: All nodes have the same weight of 1. The total weight is $|V|$. This models a traditional network with homogeneous nodes.

HALF: The first half of nodes has a weight of 2 and the second half of nodes has a weight of 1. The total weight is $\frac{3}{2}|V|$. This models a network which is a combination of two networks. The nodes of one network are twice as fast as the nodes of the other network.

SERV1: One node has a weight of $|V|+1$ and the others one of 1. The total weight is $2|V|$. This models a network with one server of high weight and with all other nodes of equally small weight. For PATH64 and GRID8x8, one node with the smallest degree is chosen as the server node.

Table 1 lists the number m of distinct eigenvalues of LC^{-1} for our example networks with different weights C. Additionally, λ_2 and λ_m, as well as α_{opt}, β_{opt} and γ are presented. Here, the eigenvalues correspond to LC^{-1}, because we do not consider different values for the edges of the network.

Table 1: The number m of distinct eigenvalues of LC^{-1} from networks with different weights C. Furthermore, λ_2, λ_m, the optimal values $\alpha_{opt} = \frac{2}{\lambda_2+\lambda_m}$, $\beta_{opt} = \frac{2}{1+\sqrt{1-\gamma^2}}$ and the value of $\gamma = \frac{\lambda_m-\lambda_2}{\lambda_m+\lambda_2}$ are displayed.

Network	Capacity	m	λ_2	λ_m	α_{opt}	β_{opt}	γ
PATH64	HOMO	64	0.002409	3.997591	0.500000	1.906455	0.998795
	HALF	64	0.001750	3.990781	0.500937	1.919639	0.999123
	SERV1	64	0.001015	3.997553	0.500179	1.938233	0.999492
GRID8x8	HOMO	33	0.152241	7.695518	0.254850	1.567586	0.961201
	HALF	64	0.093933	7.401866	0.266816	1.636018	0.974937
	SERV1	59	0.024937	7.695057	0.259068	1.796160	0.993540
HYP6	HOMO	7	2.000000	12.000000	0.142857	1.176571	0.714286
	HALF	12	1.219224	11.089454	0.162487	1.251980	0.801892
	SERV1	12	0.151868	11.922994	0.165633	1.635482	0.974846

The optimal value α_{opt} is used for the FOS, and the optimal value β_{opt} is used for the SOS. It is always $m = 64$ for the path, because it has a diameter of 63. It is obvious that $m-1$ cannot be less than the diameter for any weight. The network GRID8x8 has a diameter of 14 and m varies between 33 and 64. For the networks GRID8x8 and HYP6, m is always larger for the weights HALF and SERV1 than for HOMO.

The characteristics mentioned so far are independent of the load distribution. We choose the following initial load distribution for our balancing schemes. They model a highly unbalanced as well as a slightly unbalanced load distribution.

PEAK: One node has a load of $100|V|$ and the others have a load of 0.

RAN: $100|V|$ load items are randomly distributed among the nodes.

We compare the FOS, SOS and OPT schemes for various networks and weights. We use the optimal parameter $\alpha_{opt} = \frac{2}{\lambda_2+\lambda_n}$ for the FOS. Additionally, we use the optimal parameter $\beta_{opt} = \frac{2}{1+\sqrt{1-\gamma^2}}$. Both are presented in Table 1. Additionally, Table 1 lists the value γ which displays the rate of convergence for FOS and SOS. It can be observed that γ always increases in the following order: HOMO, HALF and SERV1.

We can choose an arbitrary ordering of the eigenvalues for the scheme OPT, but it is possible to trap in numerically unstable conditions for certain orderings [EFMP99]. We choose the order by the Leja Points [Rei91]. The eigenvalues $\lambda_2, \ldots, \lambda_m$ are sorted to a new sequence $\bar{\lambda}_2, \ldots, \bar{\lambda}_m$, such that $\bar{\lambda}_2 = \lambda_m$ and $\bar{\lambda}_i$, $i = 3, \ldots, m$, with $\prod_{j=2}^{i-1} \left|1 - \frac{\bar{\lambda}_i}{\bar{\lambda}_j}\right| \bar{\lambda}_i$ maximal.

The balancing process stops after t iterations when the error $||w^t - \overline{w}||_2$ is less than 0.01. Although the optimal scheme OPT results in an error of 0.0 after $m-1$ iterations, the error often gets below 0.01 earlier than that.

As it was demonstrated in the previous section, all schemes calculate the unique l_2-minimal flow. Table 2 presents the characteristics of the l_2-minimal flows. The flow for the heavily unbalanced load distribution PEAK is much higher than for the slightly unbalanced load distribution RAN. With respect to the weights, the flow is decreasing in the order HOMO, HALF and SERV1 for the load PEAK and decreasing in the order SERV1, HALF and HOMO for the load RAN. Furthermore, the flow values for the weight SERV1 are almost equal for the load distributions PEAK and RAN. This is true due to the fact that for PEAK, half of the total load has to be moved from the overloaded node to all other nodes. On the other hand, for RAN the node with the highest weight has a load that is about equal to all other nodes and must get about half of the total load from all other nodes.

Table 2: The l_2-minimal flows and the number of iterations for each scheme.

Network	Load	Capacity	Quality of l_2-minimal flow			iterations		
			$l_1 = \sum_e \lvert x_e\rvert$	$l_2 = \sqrt{\sum_e x_e^2}$	$l_\infty = \max_e\{\lvert x_e\rvert\}$	FOS	SOS	OPT
PATH64	PEAK	HOMO	201600	29214	6300	9655	294	63
		HALF	167465	25676	6266	13092	340	63
		SERV1	100800	14607	3150	24153	473	63
	RAN	HOMO	1485	253	78	5831	194	63
		HALF	35544	5121	1104	11559	305	63
		SERV1	102211	14732	3147	24174	473	63
GRID8x8	PEAK	HOMO	44800	6849	3150	303	52	27
		HALF	40533	6625	3150	470	65	39
		SERV1	22400	3425	1575	1945	136	33
	RAN	HOMO	573	66	18	189	35	22
		HALF	4749	658	144	339	49	34
		SERV1	22383	3428	1574	1946	136	35
HYP6	PEAK	HOMO	19200	2844	1050	37	18	6
		HALF	18267	2813	1050	56	22	11
		SERV1	9600	1422	525	497	69	11
	RAN	HOMO	416	37	8	23	12	6
		HALF	1450	199	39	38	16	10
		SERV1	9557	1421	526	497	69	11

Table 2 lists the number of iterations that are needed for the schemes FOS, SOS and OPT. Obviously, FOS needs many more iterations than SOS and SOS needs more iterations than OPT. For FOS and SOS, the load imbalance is a major factor for the number of iterations, whereas OPT always results in a balanced load after $m-1$ iterations. We can observe that OPT often finishes after less than $m-1$ iterations if the error is already below 0.01.

Figures 1 and 2 display the relation between the error and the number of iterations for the network GRID8x8. Figure 1 presents the load balancing schemes FOS and OPT that are started on the initial load distribution PEAK and Figure 2 that are started on the initial load distribution RAN.

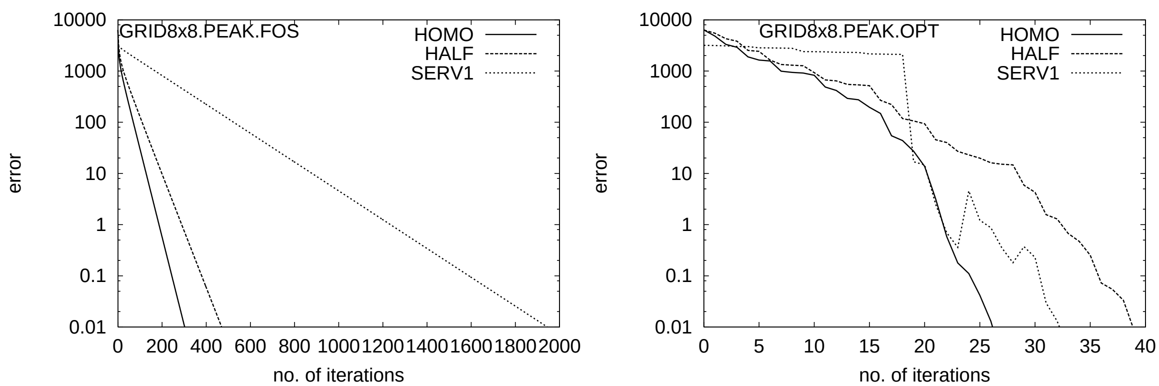

Figure 1: Balancing load PEAK on GRID8x8 with schemes FOS (left) and OPT(right).

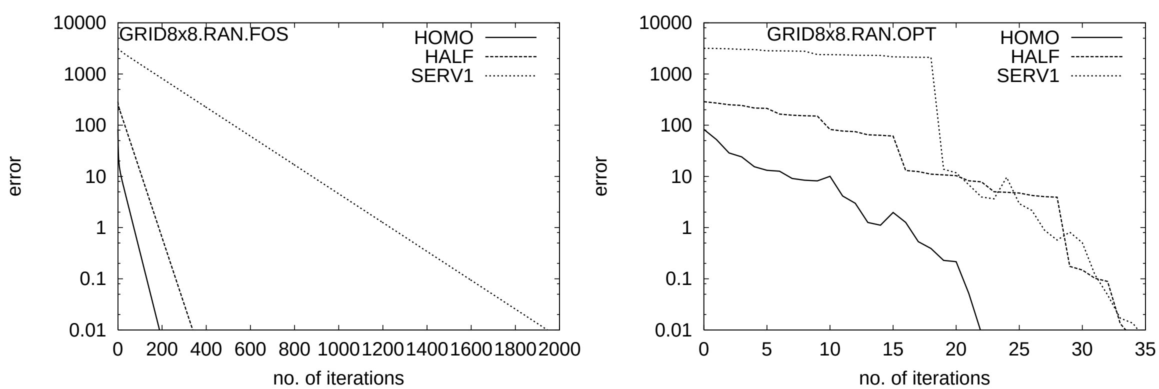

Figure 2: Balancing load RAN on GRID8x8 with schemes FOS (left) and OPT(right).

It is obvious that load balancing on heterogeneous networks can lead to high balancing flows, especially when the weights of the network are very unbalanced. Nevertheless, the algorithms calculate the l_2-minimal flow which keeps the flow migration low. Furthermore, the number of iterations also increases, but it increases for FOS and SOS much more than for OPT. Additionally, OPT has a maximum number of $m-1 < n$ iterations.

Our experiments show that it is technically easy to modify the existing diffusion load balancing schemes in order to work on heterogeneous networks.

References

[Boi90] J.E. Boillat. Load balancing and poisson equation in a graph. *Concurrency - Practice & Experience*, 2:289–313, 1990.

[Che70] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in analysis*, pages 195–199, 1970.

[Cyb89] G. Cybenko. Load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7:279–301, 1989.

[DFM99] R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. *Parallel Computing*, 25(7):789–812, 1999.

[DK86] J. Dodziuk and W.S. Kendall. Combinatorial laplacians and isoperimetric inequality. *Pitman Res. Notes Math. Ser.*, pages 68–74, 1986.

[DMN97] R. Diekmann, S. Muthukrishnan, and M.V. Nayakkankuppam. Engineering diffusive load balancing algorithms using experiments. In G. Bilardi et al., editor, *IRREGULAR'97*, LNCS 1253, pages 111–122, 1997.

[DMP00] T. Decker, B. Monien, and R. Preis. Towards optimal load balancing topologies. In *Euro-Par 2000 Parallel Processing*, LNCS 1900, pages 277–287, 2000.

[DS91] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of markov chains. *The Annals of Applied Probability*, 1:36–61, 1991.

[EFMP99] R. Elsässer, A. Frommer, B. Monien, and R. Preis. Optimal and alternating-direction loadbalancing schemes. In P. Amestoy et al., editor, *EuroPar'99*, LNCS 1685, pages 280–290, 1999.

[EMP00] R. Elsässer, B. Monien, and R. Preis. Diffusive load balancing schemes on heterogeneous networks. In *12th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 30–38, 2000.

[FWM94] G. Fox, R. Williams, and P. Messina. *Parallel Computing Works!* Morgan Kaufmann, 1994.

[GMS98] B. Ghosh, S. Muthukrishnan, and M.H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31(4):331–354, 1998.

[GV61] G. Golub and R. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods. In *Numer. Math.*, pages 147–156, 1961.

[HB99] Y.F. Hu and R.J. Blake. An improved diffusion algorithm for dynamic load balancing. *Parallel Computing*, 25(4):417–444, 1999.

[HBE98] Y.F. Hu, R.J. Blake, and D.R. Emerson. An optimal migration algorithm for dynamic load balancing. *Concurrency: Practice & Experience*, 10(6):467–483, 1998.

[Moh89] B. Mohar. Isoperimetric numbers of graphs. *J. Combin. Theory*, 47(3):274–291, 1989.

[Rei91] L. Reichel. The application of leja points to richardson iteration and polynomial preconditioning. *Linear Algebra and its Applications*, 154-156:389–414, 1991.

[SG99] V.S. Sunderarm and G.A. Geist. Heterogeneous parallel and distributed computing. *Parallel Computing*, 25:1699–1721, 1999.

[Sin93] A. Sinclair. *Algorithms for Random Generation and Counting.* Birkhäuser, 1993.

[SJ89] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inform. and Comput.*, 82:93–133, 1989.

[Var62] R. Varga. *Matrix Iterative Analysis.* Prentice-Hall, 1962.

[XL97] C. Xu and F.C.M. Lau. *Load Balancing in Parallel Computers.* Kluwer, 1997.