Alcom-FT Technical Report Series
ALCOMFT-TR-03-151

DIFFUSION LOAD BALANCING IN STATIC AND DYNAMIC NETWORKS *

- A Survey -

R. Elsässer
University of Paderborn
Fürstenallee 11
33102 Paderborn, Germany
elsa@uni-paderborn.de

B. Monien
University of Paderborn
Fürstenallee 11
33102 Paderborn, Germany
bm@uni-paderborn.de

Abstract The problem of balancing dynamically generated work load among the processors of a parallel or distributed system occurs in a wide range of applications. Several load balancing methods have been developed and analyzed. In this paper, we consider the load balancing method called diffusion, which belongs to the class of nearest neighbor load balancing algorithms.

First, we present several diffusion schemes, we describe their convergence rates, and analyze the balancing flow computed by these schemes. Due to the local behavior of diffusion, these schemes are especially well-suited for large networks. Furthermore, we show that simple schemes can be applied directly to dynamic networks, too. We also present a somehow related load balancing algorithm, which solves the so-called token distribution problem on static and dynamic networks, and compare the behavior of this algorithm with the behavior of diffusion. We conclude with a discussion about new strategies which improve the performance of simple diffusion schemes on static networks.

*This work was partially supported by the German Research Association (DFG) within the SFB 376 “Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen” and by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

1. Introduction

Load balancing is a very important prerequisite for the efficient use of parallel and distributed systems. Many parallel applications have work loads with a dynamic behavior, i.e., they can change dramatically during runtime. Therefore, the load has to be balanced in order to achieve an efficient use of the processor network. Obviously, we can ensure an overall benefit of the computation only if the balancing scheme itself is highly efficient.

Examples showing the importance of efficient load balancing strategies are parallel applications that dynamically generate a large number of jobs on the processors of a distributed system. Especially in grid or cluster computing, the problem of unbalanced load in large network topologies can occur. In order to minimize the computation time, we must redistribute the jobs among the nodes of the network so that a balanced situation is obtained. In very large networks arising in grid or cluster computing, it is important to use load balancing algorithms which act in a local manner, i.e., the nodes only access the load information of their direct neighbors.

In this paper, we model the load balancing problem as follows. The input is a network with n processors, each of them containing some work load. The network is described by a graph $G = (V, E)$, and the load of node $i \in V$ is represented by w_i. We assume that each processor can only access the load information of its direct neighbors. Now, the goal is to balance the load by transmitting load along edges, in iterations, so that after termination each node i has the load

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

Highly popular classes of load balancing algorithms are the so called diffusion schemes (e.g. Cybenko, 1989; Boillat, 1990; Muthukrishnan et al., 1998; Diekmann et al., 1999). There, the flow moved along each edge is dependent on the gradient in the loads across it. These schemes are performed locally, i.e., each step of the algorithm can be implemented in a local manner so that each node consults only its neighbors and avoids expensive global communication.

Another well-known local iterative loadbalancing method is the so-called dimension exchange (e.g. Cybenko, 1989; Xu and Lau, 1997; Arndt, 2001; Arndt, 2002). This method differs from diffusion by its communication ability in the network. While diffusion algorithms assume that a processor can send and receive messages from to/from all of its neighbors simultaneously, the dimension exchange approach is more restrictive and allows each processor to communicate with one of its neighbors during an iteration.

It is known that a strong relationship between the convergence rates of these algorithms and the spectral properties of the underlying graphs exists (e.g. Cybenko, 1989; Diekmann et al., 1999; Muthukrishnan et al., 1998; Xu and Lau, 1997). By increasing the condition number of some associated matrices, the convergence of most diffusion schemes as well as of dimension exchange also increases. In the following sections, we analyze the convergence rate of all known diffusion schemes and provide strategies on how to improve them for several graph classes and interconnection topologies.

In Elsässer et al., 2003, we showed that simple diffusion schemes can be generalized to dynamic networks. In dynamic networks, the set of edges may vary at each time step, while the set of vertices is fixed during the iterations. By using the methods of Bertsekas and Tsitsiklis, 1989, it can be easily shown that the dimension exchange scheme introduced in Cybenko, 1989 is also stable on dynamic networks. Nevertheless, we were not able to show that more efficient diffusion schemes converge on dynamic networks.

If we consider static networks, it is advisable to use load balancing algorithms which act in two phases. The first phase answers the question "how much" load has to be sent over the edges of the graph in order to obtain a balanced load situation, i.e., it determines the *balancing flow*. The second phase computes a scheduling of the load items according to the balancing flow computed in the first phase. If the current load of each processor is higher than the total load that has to be sent to its neighbors, then the processors can balance their loads in one step and no scheduling is needed. However, in most cases this condition does not hold and therefore, we use a scheduling phase to decide in which order the load has to be moved.

The balancing flow can be computed by solving a system of linear equations (see Hu et al., 1998). In Hu and Blake, 1999, the authors described an algorithm for computing the balancing flow, in which first the vector $\overline{w}$ is determined and then the system of linear equation is solved in parallel. In this paper, we consider algorithms which are executed in a completely distributed and local manner, so that the balancing flow and the vector $\overline{w}$ are computed simultaneously.

The problem of finding an optimal schedule for the load items according to some balancing flow calculated a-priori is of interest in a much border context than load balancing. Some general lower bounds for the number of steps needed to schedule a balancing flow is given in

Diekmann et al., 1999. Additionally, the authors analyzed there several known greedy strategies for this scheduling problem.

The diffusion schemes we consider in this paper were introduced for homogeneous networks, i.e., each processor of the network has the same performance. However, all of these algorithms work well on heterogeneous networks, too (see Diekmann et al., 1999; Elsässer et al., 2002a). Heterogeneous networks are modelled by (node and/or edge) weighted graphs, where the performance of the processors and of the links in the system are represented by the weights of the nodes and edges, respectively.

Somehow related approaches to local iterative load balancing algorithms are the methods described in Aiello et al., 1993; Ghosh et al., 1999, which were developed to solve the so-called token distribution problem. There, the authors stated a relationship between the convergence rate of these algorithms and the edge and node expansion of the underlying processor network. These algorithms do not completely balance the load situation, however, they can be applied as well to static as to dynamic networks. Note that, in contrast to the algorithmic model described above, the token distribution problem handles indivisible unit size tokens and only allows one token to be sent over an edge during an iteration step.

The outline of the paper is as follows. In the next section, we describe the general diffusion scheme introduced in Cybenko, 1989, we analyze the convergence of this algorithm and show how to generalize it to dynamic networks. We also describe a simple algorithm which solves the load balancing problem in a restricted model, and compare this algorithm with diffusion. In Section 3, we present several improved diffusion schemes, we describe their performance on homogeneous and heterogeneous systems, and we analyze the flow computed by these algorithms. We conclude by referring to some related load balancing strategies, improvement possibilities and further research directions.

2. Simple Diffusion Schemes on Static and Dynamic Networks

As described in the introduction, we first consider the general diffusion scheme introduced in Cybenko, 1989 on static and dynamic topologies. Let $G = (V, E)$ be a graph, and let w_i^k denote the load after the k^{th} iteration step on node $i \in V$. The general diffusion scheme on a static network represented by the graph G has the form:

$$w_i^k = w_i^{k-1} - \sum_{\{i,j\} \in E} \alpha_{i,j}(w_i^{k-1} - w_j^{k-1}), \qquad (1)$$

where $\alpha_{i,j}$ is a properly chosen constant ensuring convergence as described below. The term $|\alpha_{i,j}(w_i^{k-1} - w_j^{k-1})|$ is the amount of load sent over the edge (i,j) in iteration step k. It is obvious that the flow moved over an edge depends on the load of its incident nodes, and therefore, this scheme is performed locally with all nodes consulting their direct neighbors only. Equation (1) can be written in matrix notation as $w^k = Mw^{k-1}$, where $M = I - \tilde{L}$ and $\tilde{L}$ is the weighted Laplacian of G which is defined as follows: $\tilde{L}$ contains $\alpha_{i,j}$ at position (i,j), and the weighted node degree $\sum_{\{i,j\}\in E} \alpha_{i,j}$ as diagonal entry. For the unweighted Laplacian L of G holds $\alpha_{i,j} = 1$ for any $(i,j) \in E$, and $\alpha_{i,j} = 0$, $i \neq j$ otherwise.

We call $e^k = w^k - \overline{w}$ the *error* after iteration k. As shown in e.g. Diekmann et al., 1999, equation (1) implies that $||e^k||_2 \leq \max\{|\mu_2|, |\mu_n|\} \cdot ||e^{k-1}||_2$, where μ_2 and μ_n are the second largest and smallest eigenvalues of M, respectively. Now, the parameters $\alpha_{i,j}$s has to be chosen such that $1 = \mu_1 \geq \mu_2 \geq \cdots \geq \mu_n > -1$. The eigenvalue $\mu_1 = 1$ is simple with eigenvector $(1,\ldots,1)$ iff G is connected. We denote by $\gamma = \max\{|\mu_2|, |\mu_n|\}$ the second largest eigenvalue of M according to absolute values, and call it the *diffusion norm* of G. The matrix M is symmetric and doubly stochastic, i.e., its rows and columns sum up to 1. We call such a matrix a *diffusion matrix.*

It is often convenient to set $\alpha_{i,j} = \alpha$ for any $(i,j) \in E$. Then, $M = I - \alpha L$ and γ is minimized for $\alpha = \frac{2}{\lambda_2+\lambda_n}$ (e.g. Xu and Lau, 1997), where we denoted by λ_i the i^{th} eigenvalue of L.

Since $\overline{w} = \frac{\sum_{i=1}^{n} w_i^0}{n}(1,\ldots,1)$ and $w^0 = \sum_{i=1}^{n} z_i$ (Hu and Blake, 1999; Diekmann et al., 1999), where z_i is a (not necessarily normalized) eigenvector of M corresponding to μ_i, it holds that

$$||e^k||_2 \leq \gamma ||e^0||_2. \tag{2}$$

Now, we call the system ϵ-balanced if $\frac{||e^k||_2}{||e^0||_2} \leq \epsilon$. As shown in Muthukrishnan et al., 1998, the system is ϵ-balanced using the general diffusion scheme of Cybenko in

$$k = O\left(\frac{\log(1/\epsilon)}{1-\gamma}\right) \tag{3}$$

steps. Setting $\alpha = \frac{2}{\lambda_2+\lambda_n}$, equation (3) results in

$$k = O\left(\frac{\log(1/\epsilon)\cdot d_{\max}}{\lambda_2}\right),$$

where $d_{\max}$ is the maximum vertex degree in G.

Now we turn our attention to dynamic networks. Dynamic networks are described by a sequence of undirected graphs $(G_k = (V, E_k))_{k\geq 0}$, where every node $i \in V$ contains the initial load w_i. Here G_k represents the network with edge set E_k which occurs at time step $k \geq 0$. We assume that each node knows which of its incident edges are present at each iteration step.

The diffusion scheme defined in equation (1) can be directly generalized to dynamic networks. However, the nodes do not know the eigenvalues of G_k at each step, and therefore, the values $\alpha_{i,j}$ have to be chosen by using local information only. We set $\alpha_{i,j}^k = \frac{1}{c \cdot \max\{d_i^k, d_j^k\}}$ at time step k, where $c \in (1, 2]$ is a constant and d_i^k represents the degree of node i at time k. By using these values for $\alpha_{i,j}$, we obtain the weighted Laplacian $\tilde{L}_k$ of G_k. Then, the diffusion matrix M_k in iteration step k has the form $M_k = I - \tilde{L}_k$ and $w^{k+1} = M_k w^k$. Therefore, the convergence rate at time step k is determined by the diffusion norm γ_k of G_k and it can be shown that the convergence of the diffusion algorithm depends on the average value $\frac{\sum_{i=0}^{k-1}(\lambda_2^i/d_{\max}^i)}{k}$, where λ_2^i and $d_{\max}^i$ are the second smallest eigenvalue of the Laplacian L_i and the maximum vertex degree of G_i, respectively, which occur in iteration i.

The following theorem provides an asymptotic exact result on the convergence rate of the diffusion algorithm (see Elsässer et al., 2003).

THEOREM 1 *Let $(G_k)_{k\geq 0}$ be a sequence of graphs and denote with $A_K = \frac{\sum_{k=0}^{K-1} \lambda_2^k/d_{max}^k}{K}$ the average value of λ_2^k/d_{max}^k occurring during the first K iterations. The diffusion scheme defined in equation (1) needs at most K steps to ϵ-balance the system, where*

$$K = O\left(\frac{1}{A_K} \cdot \ln(1/\epsilon)\right).$$

This theorem implies that the graphs G_k do not always have to be connected during the iterations to obtain convergence. Note that we used in the analysis of this algorithm some global properties of the graph such as λ_2^k and $d_{\max}^k$. However, during the execution of the algorithm any node requires local information only, and hence, expensive global communication can always be avoided.

If $\lambda_2^i/d_{\max}^i$ is always larger than some value λ/d, Theorem 1 results in the following corollary.

COROLLARY 2 *Let $(G_k)_{k \geq 0}$ be a sequence of graphs. If $\frac{\lambda_2^k}{d_{max}^k} \geq \frac{\lambda}{d}$ for $k \geq 0$, then the diffusion scheme defined in equation (1) needs at most*

$$K = O\left(\frac{d}{\lambda} \cdot \ln(1/\epsilon)\right)$$

iteration steps to ϵ-balance the system.

For diffusion algorithms, we assume that the load items can be arbitrarily divided and any amount of load can be moved over the edges of the graph. In a related model, namely token distribution, it is assumed that any node has an initial load consisting of indivisible unit size tokens, and only one token can be sent over an edge during an iteration step. Simple algorithms for solving the load balancing problem in such a restricted model have also been developed and analyzed. Then, these algorithms have been generalized to dynamic and asynchronous networks (see Aiello et al., 1993; Ghosh et al., 1999).

In these papers, the authors presented several algorithms based on different communication models of the network. We only analyze the algorithm which assumes that each node can communicate with all of its neighbor in one iteration step.

First, we consider static networks. Let $G = (V, E)$ be a graph. Then, at each step k, a token is sent from node i to node j across edge (i, j) if at the beginning of step k node i contains at least $2d_{\max} + 1$ more tokens than j.

Let $i(G) = \min_{\substack{X \subset V \\ |X| \leq |V|/2}} \frac{E(X,\overline{X})}{|X|}$ the edge expansion of G, where $E(X, \overline{X})$ represents the number of edges between X and $\overline{X}$. As shown in Ghosh et al., 1999, the algorithm described above balances the load with initial imbalance Δ in G in $O\left(\frac{\Delta}{i(G)}\right)$ steps to within $O\left(\frac{d_{\max}^2 \log(n)}{i(G)}\right)$ tokens. We call a system balanced to within t tokens, whenever the maximum difference in the number of tokens between any two nodes in the graph is at most t.

The convergence of the algorithm can also be expressed in terms of node expansion. The node expansion $n(G)$ of G is defined as $n(G) = \min_{\substack{X \subset V \\ |X| \leq |V|/2}} \frac{N(X)}{|X|}$, where $N(X)$ represents the neighbors of the vertices of X in $\overline{X}$. Then, the system is balanced to within $O\left(\frac{d_{\max} \log(n)}{n(G)}\right)$ tokens in $O\left(\frac{\Delta}{n(G)}\right)$ steps. The bound above is tight, and, depending on G, it can be very expensive to balance the load in the system to within $2d_{\max}$ tokens.

Now, if we consider dynamic networks described by a sequence of graphs $(G_k = (V, E_k))_{k\geq 0}$, it can be shown that the algorithm described above also works on such networks. Then, in every step k of the algorithm, each node i performs the following operations:

First, for each neighbor j such that $(i,j) \in E_k$, if $w_i^{k-1} - e_i^{k-1}(j) > 12d_{\max}$, i sends a message consisting of w_i^{k-1} and a token; otherwise, i sends a message consisting only of w_i^{k-1}. Here, $e_i^{k-1}(j)$ represents the estimate of the number of tokens at j for the neighbor j of i. This estimate is obtained in some previous communication step of node i with node j. Next, w_i^{k-1} is decreased by the number of tokens sent.

Second, for each message received from a neighbor j, $e_i^{k-1}(j)$ is updated according to the message, and w_i^{k-1} is updated according to the number of tokens received in order to obtain w_i^k.

This algorithm balances the load on a sequence of graphs $(G_k)_{k\geq 0}$ with initial imbalance Δ to within $O\left(\frac{d_{\max}^2 \log(n)}{i(G)}\right)$ tokens in $O\left(\frac{\Delta}{i(G)}\right)$ steps, where $d_{\max}$ is the maximal vertex degree and $i(G)$ is the worst edge expansion, which occur during the iterations.

It is worth mentioning that the algorithm described in the previous paragraphs makes use of the global term $d_{\max}$. Therefore, this algorithm can be applied to networks, in which some edges can disappear due to failures in some fixed network, or to asynchronous networks, in which the different latecies are modelled by edge failures in some original graph.

Comparing the results of Theorem 1 with the results described above, we can observe that , in contrast to the algorithm of Ghosh et al., 1999, the general diffusion scheme also converges if the graphs are often unconnected, and the convergence rate depends on the average value of $\lambda_2^k / d_{\max}^k$ occurring during the iterations. However, since $\frac{i^2(G)}{2d_{\max}} \leq \lambda_2 \leq 2i(G)$ (see e.g. Mohar, 1988; Mohar, 1989; Chung, 1997), the algorithms of Ghosh et al., 1999 can perform better (depending on G and Δ) than the general diffusion scheme. In addition, by using the token distribution algorithm each node requires only an estimate of the load of its neighbors, while by using diffusion the nodes need the exact load information of their neighbors in each iteration step.

3. Improved Polynomial Diffusion Schemes

As described in the introduction, the general diffusion scheme of Cybenko can be significantly improved by using several techniques known from linear algebra. One of these techniques makes use of the idea of over-relaxation, improving the general diffusion scheme by an almost quadratic factor. We conclude this section with the flow analysis of the polynomial diffusion schemes. In the sequel we assume that the load items are arbitrary divisible and any amount of load can be sent over the edges of the graph.

Before we start with the analysis of different diffusion schemes, we define the so-called polynomial based load balancing scheme. These schemes are defined on static networks only, since we could not generalize them to dynamic topologies.

DEFINITION 3 *Let $G = (V, E)$ be a graph and let M be a corresponding diffusion matrix with diffusion norm γ. A* polynomial based *load balancing scheme is any scheme for which the work load w^k in step k can be expressed in the form*

$$w^k = p_k(M) w^0 \quad \textit{where } p_k \in \overline{\Pi}_k \ . \tag{4}$$

Here, $\overline{\Pi}_k$ denotes the set of all polynomials p of degree $\deg(p) \leq k$ satisfying the constraint $p(1) = 1$.

The condition $p_k(1) = 1$ implies that all row sums in the matrix $p_k(M)$ are equal to 1, i.e. $\sum_{i=1}^{n} w_i^k = \sum_{i=1}^{n} w_i^0$.

The convergence of a polynomial based scheme will depend on whether (and how fast) the 'error' $e^k = w^k - \overline{w}$ will tend to zero. Using the results of e.g. Diekmann et al., 1999, the error e^k of any polynomial based scheme satisfies

$$\|e^k\|_2 \leq \max_{i=2}^{m} |p_k(\mu_i)| \cdot \|e^0\|_2 \ . \tag{5}$$

From equation (5) it follows that the general diffusion scheme of Cybenko converges due to the inequality (2). In the sequel, we call the scheme introduced in equation (1) the *first order scheme* (FOS).

A direct improvement of FOS using the idea of over-relaxation leads to the following scheme called *second order scheme* SOS (Muthukrishnan et al., 1998):

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

Here, β is a fixed parameter. This scheme is known as the second order Richardson method in numerical analysis. Investigations in Golub and Varga, 1961 show that this iteration converges to $\overline{w}$ whenever $\beta \in (0, 2)$ and that the fastest convergence occurs for $\beta = 2/\left(1 + \sqrt{1-\gamma^2}\right)$. In this case, one has (see Golub and Varga, 1961; Varga, 1962)

$$\|e^k\|_2 \ \leq \ (\beta - 1)^{\frac{k}{2}} \left(1 + k\sqrt{1-\gamma^2}\right) \cdot \|e^0\|_2 \ . \tag{7}$$

A result of Muthukrishnan et al., 1998 implies that a system is ϵ-balanced using SOS in

$$k = O(\frac{\log(1/\epsilon)}{\sqrt{1-\gamma}})$$

steps. Comparing the convergence of SOS with the convergence of FOS it turns out that SOS is faster than FOS by almost a quadratic factor.

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

$$\beta_1 = 1, \quad \beta_2 = \frac{2}{2-\gamma^2}, \quad \beta_k = \frac{4}{4-\gamma^2\beta_{k-1}}, \qquad k = 3, 4, \dots \ . \tag{8}$$

Then,

$$\|e^k\|_2 \ \leq \ (\beta-1)^{\frac{k}{2}} \frac{2}{1+(\beta-1)^k} \cdot \|e^0\|_2 \ . \tag{9}$$

The factor in (9) is always smaller than in (7) which shows that the Chebyshev method is usually to be preferred over the SOS scheme. Asymptotically, however, both methods can be regarded to perform identically (see Hu and Blake, 1999; Diekmann et al., 1999).

The previous schemes converge to the balanced load situation due to some convergence rate. This rate of convergence is based on the condition number of the corresponding Laplacians. If the eigenvalues of a graph are known in advance, then we can use a so-called optimal polynomial scheme, which balances the load completely in $m-1$ iteration steps, where m is the number of distinct eigenvalues of L. We denote by $\tilde{\lambda}_i$, $i \in \{1, \dots, m\}$ the m distinct eigenvalues of L. Due to the simple optimal scheme mentioned above, node i sends in iteration step k the flow $\frac{1}{\tilde{\lambda}_{k+1}}(w_i^{k-1} - w_j^{k-1})$ over the edge (i,j) (Elsässer et al., 1999). Then w^k satisfies the equation

$$w_i^k = w_i^{k-1} - \sum_{(i,j)\in E} \frac{1}{\tilde{\lambda}_{k+1}}(w_i^{k-1} - w_j^{k-1}).$$

Therefore,

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

and for the error e^{m-1} we obtain after $m-1$ iterations that

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

For each iteration a different eigenvalue of L is used, eliminating the corresponding eigenvectors to this eigenvalue at a time.

The order of the eigenvalues may be arbitrary, however, it influences the numerical stability of the calculation. In our experiments, we tested this optimal scheme called OPT using several orderings on the distinct eigenvalues of plenty of graphs. The best results were obtained by using an ordering based on weighted Leja points (see Reichel, 1991). In Diekmann et al., 1999, an improvement of this scheme is presented, which also completely balances the load in $m-1$ steps, however, this algorithm is stable from the numerical point of view.

The optimal polynomial scheme presented above motivates the construction of sparse graph classes with a small number of different eigenvalues. These networks are well-suited as virtual topologies in a bus system, where we allow each processor to communicate with only a restricted number of other processors in the system (in order to avoid high communication costs). If these virtual topologies have a small number of distinct eigenvalues, then the optimal polynomial scheme balances the load in the network in only a few number of iteration steps. Discussions and an analysis on such well-connected interconnection topologies can be found in Decker et al., 2000 and in Elsässer et al., 2001.

Let us now consider the flow computed by the polynomial based diffusion schemes. We already mentioned in the introduction that the balancing flow can be computed by solving a system of linear equations (see e.g. Hu et al., 1998). Let $A \in \{-1, 0, 1\}^{n \times N}$ be the incidence matrix of G, where $N = |E|$. A balancing flow x is any flow with $Ax = w^0 - \overline{w}$. Now, in Diekmann et al., 1999; Hu and Blake, 1999 the following theorem has been shown.

THEOREM 4 *Let $p_k \in \overline{\Pi}_k$ be a polynomial satisfying the 3-term recurrence equation*

$$p_k(t) = (\sigma_k t - \tau_k) p_{k-1}(t) + \rho_k p_{k-2}(t) \ , \tag{10}$$

with

$$\sigma_k - \tau_k + \rho_k = 1 \quad \forall \ k = 1, 2, \ldots . \tag{11}$$

Then the polynomial diffusion scheme $w^k = p_k(M) w^0$ computes an l_2 minimal flow x on the edges of G among all balancing flows y with the property $Ay = w^0 - \overline{w}$.

This result can easily be generalized to edge weighted graphs by considering the weighted l_2-norm of a flow x, i.e., $||x||_2 = \left(\sum_{i=1}^{N} c_i (x_i)^2 \right)^{1/2}$, where we denoted by c_i the weight of edge i.

The algorithms and results described above can also be generalized to node weighted graphs. Let $c_1, c_2, \ldots, c_n$ be the weights of the nodes in a graph $G = (V, E)$ and let $C \in I\!R^{n \times n}$ denote the weight matrix, in which in the diagonal we have the values c_i and 0 elsewhere. Node weighted graphs describe heterogeneous processor networks, where the weights of the nodes represent different processor speeds. In such cases, the load in the system has to be balanced among the processors proportionally to their computational power. Formally, we are interested in a balancing flow over the edges of the network such that, after termination, each node i has the balanced proportional work load of $\tilde{w}_i := \frac{\sum_{i=1}^n w_i}{\sum_{i=1}^n c_i} c_i$, i.e., the time to consume the load on each node is equal.

In Elsässer et al., 2002a, the following lemma was shown.

Lemma 5 *Let $G = (V, E)$ be a (weighted) graph and let L be its Laplacian. Furthermore, let $c_1, \ldots, c_n$ be the weights of the vertices in G. The eigenvalues of the matrix LC^{-1} are real and non-negative. Moreover, the eigenvectors of LC^{-1} form a basis in $I\!R^n$.*

Since $\tilde{w}$ is an eigenvector of LC^{-1} corresponding to 1, the result of Lemma 5 implies that for any polynomial based diffusion scheme described by some polynomial p holds:

$$||e^k||_2 \quad \leq \quad \sqrt{\frac{c_{\max}}{c_{\min}}} p_k(\gamma) ||e^0||_2.$$

Here, $c_{\max}$ and $c_{\min}$ represent the maximal and minimal value, respectively, of the node weights in the graph. Therefore, the polynomial based diffusion schemes FOS, SOS and the Chebyshev scheme can be directly generalized to heterogeneous processor networks, where the convergence differs only by the constant factor $\sqrt{\frac{c_{\max}}{c_{\min}}}$ from the homogeneous case. The optimal schemes presented in this section can also be generalized to heterogeneous networks, where the number of total iteration steps equals the number of different eigenvalues of LC^{-1}. The flow calculated by the polynomial diffusion schemes in the case of heterogeneous networks is l_2-optimal, too.

4. Conclusion

In this paper, we described several diffusion based load balancing strategies on interconnection topologies. We analyzed their convergence rate and studied the balancing flow computed by these algorithms. We also considered the behavior of simple diffusion schemes on dynamic topologies. We compared this schemes with some related algorithms which were developed to solve the token distribution problem on dynamic networks. We have seen that in some cases, the token distribution algorithm can achieve faster a nearly balanced state than simple diffusion schemes.

It is an open problem, if more efficient diffusion schemes like SOS can also be generalized to dynamic networks. If such schemes would provide a similar improvement factor of the convergence rate on dynamic networks, diffusion would always perform better than the algorithms of Ghosh et al., 1999 on such topologies.

It is worth mentioning that the convergence rate of all known diffusion schemes can be improved on some graph classes and well-known interconnection topologies. One approach consists of the *Alternating-Direction Iterative Scheme* (ADI) introduced in Elsässer et al., 1999. This scheme converge faster than diffusion by at most a factor of 2 on Cartesian products of graphs. However, the flow computed by the ADI scheme is not minimal in the l_2-norm and may be very large if the optimal parameters w.r.t. the convergence rate are used. We can also improve the convergence for several interconnection topologies by using edge-weighted graphs. This approach leads to an improvement factor of at most 2 for grid-like structures (see Diekmann et al., 1997; Elsässer et al., 2002b).

In this paper, we assumed that the load items are arbitrary divisible. In Muthukrishnan et al., 1998; Rabbani et al., 1998, the performance of diffusion algorithms for indivisible unit size tokens has been analyzed. The authors showed that that diffusion do not completely balance the load for such settings. In order to improve the quality of the final load situation in the network, a randomized algorithm was introduced in Elsässer and Monien, 2003, which reduces the final load imbalance in l_2-norm to $O(\sqrt{n})$. This randomized algorithm is based on the simple diffusion scheme described in Section 2. It is an open problem, if the efficient diffusion schemes like SOS can also be used to develop a similar randomized algorithm with an improved convergence rate.

References

Aiello, W., Awerbuch, B., Maggs, B., and Rao, S. (1993). Approximate load balancing on dynamic and asynchronous networks. In *Proc. of STOC'93*, pages 632–641.

Arndt, H. (2001). Load balancing by finite dimension exchange algorithms. Technical report, Bergische Universität GH Wuppertal.

Arndt, H. (2002). On finite dimension exchange algorithms. Technical report, Bergische Universität GH Wuppertal.

Bertsekas, D. and Tsitsiklis, J. (1989). *Parallel and Distributed Computation, Numerical Methods.* Prentice-Hall.

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

Chung, F. (1997). *Spectral Graph Theory*, volume 92 of *CBMS Regional conference series in mathematics*. American Mathematical Society.

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

Decker, T., Monien, B., and Preis, R. (2000). Towards optimal load balancing topologies. In *Euro-Par'00*, pages 277–287.

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

Diekmann, R., Muthukrishnan, S., and Nayakkankuppam, M. (1997). Engineering diffusive load balancing algorithms using experiments. In *Solving Irregulary Structured Problems in Parallel, (IRREGULAR)*, LNCS 1253, pages 111–122.

Elsässer, R., Frommer, A., Monien, B., and Preis, R. (1999). Optimal and alternating-direction loadbalancing schemes. In *EuroPar'99*, pages 280–290.

Elsässer, R., Královič, R., and Monien, B. (2001). Scalable sparse topologies with small spectrum. In *Proc. of STACS'01*, pages 218–229.

Elsässer, R. and Monien, B. (2003). Load balancing of unit size tokens and expansion properties of graphs. In *Proc. of SPAA'03*.

Elsässer, R., Monien, B., and Preis, R. (2002a). Diffusion schemes for load balancing on heterogeneous networks. *Theory of Computing Systems*, 35:305–320.

Elsässer, R., Monien, B., Rote, G., and Schamberger, S. (2002b). Toward optimal diffusion matrices. In *Proc. of IPDPS'02*.

Elsässer, R., Monien, B., and Schamberger, S. (2003). Load balancing on dynamic networks. *manuscript*.

Ghosh, B., Leighton, F. T., Maggs, B. M., Muthukrishnan, S., Plaxton, C. G., Rajaraman, R., Richa, A., Tarjan, R. E., and Zuckerman, D. (1999). Tight analyses of two local load balancing algorithms. *SIAM Journal on Computing*, 29:29–64.

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

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

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

Mohar, B. (1988). Isoperimetric inequalities, growth, and the spectrum of graphs. *Lin. Alg. Appl.*, 103:119–131.

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

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

Rabbani, Y., Sinclair, A., and Wanka, R. (1998). Local divergence of markov chains and the analysis of iterative load-balancing schemes. In *Proc. of FOCS'98*, pages 694–703.

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

Varga, R. (1962). *Matrix Iterative Analysis*. Prentice-Hall.

Xu, C. and Lau, F. (1997). *Load Balancing in Parallel Computers*. Kluwer.