Optimal and Alternating-Direction Load Balancing Schemes

Robert Elsässer[1,⋆], Andreas Frommer[2], Burkhard Monien[1,⋆], and Robert Preis[1,⋆]

[1] Department of Mathematics and Computer Science
University of Paderborn, D-33102 Paderborn, Germany
{elsa,bm,robsy}@uni-paderborn.de
[2] Department of Mathematics and Institute for Applied Computer Science
University of Wuppertal, D-42097 Wuppertal, Germany
frommer@math.uni-wuppertal.de

Abstract. We discuss iterative nearest neighbor load balancing schemes on processor networks which are represented by a cartesian product of graphs like e.g. tori or hypercubes. By the use of the Alternating-Direction Loadbalancing scheme, the number of load balance iterations decreases by a factor of 2 for this type of graphs. The resulting flow is analyzed theoretically and it can be very high for certain cases. Therefore, we furthermore present the Mixed-Direction scheme which needs the same number of iterations but results in a much smaller flow.
Apart from that, we present a simple optimal diffusion scheme for general graphs which calculates a minimal balancing flow in the l_2 norm. The scheme is based on the spectrum of the graph representing the network and needs only $m-1$ iterations in order to balance the load with m being the number of distinct eigenvalues.

1 Introduction

We consider the load balancing problem in a synchronous, distributed processor network. Each node of the network has a computational load and, if it is not equally distributed, it will have to be balanced by moving parts of the load via communication links between two processors. We model the processor network by an undirected, connected graph $G = (V, E)$ in which node $v_i \in V$ contains a computational load of w_i. We want to determine a schedule in order to move load across edges so that finally the weight on each node is approximately equal. In each step a node is allowed to move any size of load to each of its neighbors in G. Communication is only allowed along the edges of G.

This problem models load balancing in parallel adaptive finite element simulations where a geometric space, which is discretized using a mesh, is partitioned into sub-regions. Besides, the computation proceeds on mesh elements in each

⋆ Supported by European Union ESPRIT LTR Project 20244 (ALCOM-IT) and German Research Foundation (DFG) Project SFB-376.

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 280–290, 1999.

sub-region independently (see e.g. [4, 5]). Here we associate a node with a mesh region, an edge with the geometric adjacency between two regions, and load with the number of mesh elements in each region. As the computation proceeds, the mesh gets refined or coarsened depending on the characteristics of the problem and the size of the sub–regions (numbers of elements) has to be balanced. Since elements have to reside in their geometric adjacency, they can only be moved between adjacent mesh regions, i.e. via edges of the graph [5]. In this paper we consider the calculation of the load balancing flow. In a further step, the moved load elements have to be chosen and moved to their new destination.

Scalable algorithms for our load balancing problem iteratively balance the load of a node with its neighbors until the whole network is globally balanced. The class of local iterative load balancing algorithms distinguishes between *diffusion* [2] and *dimension exchange* [2, 13] iterations. Diffusion algorithms assume that a node of the graph can send and receive messages to/from all its neighbors simultaneously, whereas dimension exchange iteratively only balances with one neighbor after the other. The *quality* of a balancing algorithm can be measured in terms of numbers of iterations that are required in order to achieve a balanced state and in terms of the amount of load moved over the edges of the graph. The earliest local method is the diffusive *first order scheme* (FOS) by [2]. It lacks in performance because of its slow convergence. With the idea of over-relaxation FOS can be sped up by an order of magnitude [6] (*second order scheme*, SOS). Diffusive algorithms have gained new attention in the last couple of years [4, 5, 7, 8, 10, 12, 13] and it has been shown that *all* local iterative diffusion algorithms determine the unique l_2-minimal flow [3].

In Sect. 3, we discuss a combination of the diffusion and the dimension-exchange model which we call the *Alternating Direction Iterative* (ADI) model. It can be used for cartesian products of graphs like tori or hypercubes, which often occur as processor networks. Here, in each iteration, a processor first communicates with its neighbors in one and then in the other direction. Thus, it communicates once per iteration with each neighbor. We show that for these graphs the upper bound on the number of load balance steps will be halved. As a drawback, we can prove that the resulting flow is not minimal in the l_2-norm and can be very large if optimal parameters for the number of iterations are used. As a (partial) remedy to this problem we present the *Mixed Direction Iterative* (MDI) scheme which needs the same number of iterations but results in a much smaller flow.

[3] introduced an optimal load balancing scheme based on the spectrum of the graph. It only needs $m - 1$ iterations with m being the number of distinct eigenvalues. This scheme keeps the load-differences small from step to step, i.e. it is numerically stable. In Sect. 4, we present a much simpler optimal scheme OPT which also needs only $m-1$ iterations. It might trap into numerical instable conditions, but we present rules of how to avoid those. The calculation of the spectrum for arbitrary large graphs is very time-consuming, but it is known for many classes of graphs and can efficiently be computed for small graphs. In Sect. 5 experiments are presented to underline the theoretical observations.

2 Definitions

Let $G = (V, E)$ be a connected, undirected graph with $|V| = n$ nodes and $|E| = N$ edges. Let $w_i \in \mathbb{R}$ be the *load* of node $v_i \in V$ and $w \in \mathbb{R}^n$ be the vector of load values. $\overline{w} := \frac{1}{n}(\sum_{i=1}^{n} w_i)(1, \ldots, 1)$ denotes the vector of the average load. Define $A \in \{-1, 0, 1\}^{n \times N}$ to be the node-edge *incidence matrix* of G. 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 which will later on be used to express the direction of the flow. Let $B \in \{0, 1\}^{n \times n}$ be the *adjacency matrix* of G. As G is undirected, B is symmetric. Column/row i of B contains 1's at the positions of all neighbors of v_i. The *Laplacian* $L \in \mathbb{Z}^{n \times n}$ of G is defined as $L := D - B$, with $D \in \mathbb{N}^{n \times n}$ containing the node degrees as diagonal entries and 0 elsewhere. Obviously, $L = AA^T$. Let $x \in \mathbb{R}^N$ be a flow on the edges of G. x is called a *balancing flow* on G iff $Ax = w - \overline{w}$. Among all possible flows we are interested in balancing flows with minimal l_2-norm defined as $\|x\|_2 = (\sum_{i=1}^{N} x_i^2)^{1/2}$. Let us also define the l_1-norm $\|x\|_1 = \sum_{i=1}^{N} x_i$ and the l_∞-norm $\|x\|_\infty = \max_{i=1}^{N} x_i$.

We consider the following *local iterative load balancing algorithm* which performs iterations on the nodes of G with communication between adjacent nodes only. This method performs on each node $v_i \in V$ the iteration

$$\forall\, e = \{v_i, v_j\} \in E:\ y_e^{k-1} = \alpha(w_i^{k-1} - w_j^{k-1}); \quad x_e^k = x_e^{k-1} + y_e^{k-1};$$
$$\text{and } w_i^k = w_i^{k-1} - \sum\nolimits_{e=\{v_i, v_j\} \in E} y_e^{k-1} \tag{1}$$

Here, y_e^k is the amount of load sent via edge e in step k with α being a parameter for the fraction of the load difference, x_e^k is the total load sent via edge e until iteration k and w_i^k is the load of node i after the k-th iteration. The iteration (1) converges to the average load $\overline{w}$ [2]. This scheme is known as the *diffusion algorithm*. Equation (1) can be written in a matrix notation as $w^k = Mw^{k-1}$ with $M = I - \alpha L \in \mathbb{R}^{n \times n}$. M contains α at position (i, j) of edge $e = (v_i, v_j)$, $1 - \sum_{e=\{v_i, v_j\} \in E} \alpha$ at diagonal entry i, and 0 elsewhere. Now let $\lambda_i(L)$, $1 \le i \le m$ be the distinct eigenvalues of the Laplacian L in increasing order. It is known that $\lambda_1(L) = 0$ is simple with eigenvector $(1, \ldots, 1)$ and $\lambda_m(L) \le 2 \cdot deg_{max}$ (with deg_{max} being the maximum degree of all nodes) [1]. Then M has the eigenvalues $\mu_i = 1 - \alpha\lambda_i$ and α has to be chosen such that $\mu_m > -1$. Note that $\mu_1 = 1$ is a simple eigenvalue to the eigenvector $(1, 1, \ldots, 1)$. Such a matrix M is called *diffusion matrix*. We denote by $\gamma = \max\{|\mu_2|, |\mu_m|\} < 1$ the second largest eigenvalue of M according to absolute values and call it the *diffusion norm* of M.

Let $w^0 = \sum_{i=1}^{m} z_i$ be represented as a sum of (not necessarily normalized) eigenvectors z_i of M with $Mz_i = \mu_i z_i$, $i = 1, \ldots, m$. Then $\overline{w} = z_1$ [3].

Definition 1. *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 $w^k = p_k(M)w^0$ where $p_k \in \overline{\Pi}_k$. Here, $\overline{\Pi}_k$ denotes the set of all polynomials p of degree* $\deg(p) \le k$ *satisfying the constraint* $p(1) = 1$.

Condition $p_k(1) = 1$ implies that all row sums in matrix $p_k(M)$ are equal to 1. For a polynomial based scheme to be feasible practically, it must be possible to rewrite it as an update process where w^k is computed from w^{k-1} (and maybe some previous iterates) involving *one* multiplication with M only. This means that the polynomials p_k have to satisfy some kind of short recurrence relation. The convergence of a polynomial based scheme depends on whether (and how fast) the 'error' $e^k = w^k - \overline{w}$ tends to zero. These errors e^k satisfy $e^k = p_k(M)e^0$ which yields $\|e^k\|_2 \leq \max_{i=2}^m |p_k(\mu_i)| \cdot \|e^0\|_2$ $k = 0, 1, \ldots$, where $e^0 = \sum_{i=2}^m z_i$ [3].

As an example, take the first order-scheme (FOS) [2], where we have $p_k(t) = t^k$. These polynomials satisfy the simple short recurrence $p_k(t) = t \cdot p_{k-1}(t)$, $k = 1, 2, \ldots,$ so that we get $w^k = Mw^{k-1}$, $k = 1, 2, \ldots$. Now $|p_k(\mu_i)| = |\mu_i^k| \leq \gamma^k$ for $i = 2, \ldots, m$ and therefore

$$\|e^k\|_2 \leq \gamma^k \cdot \|e^0\|_2 \ .$$

Here, $\gamma(M) = \max\{|1-\alpha\lambda_2(L)|, |1-\alpha\lambda_m(L)|\}$ and the minimum of γ is achieved for the optimal value $\alpha = \frac{2}{\lambda_2(L)+\lambda_m(L)}$. Then we have $\gamma = \frac{\lambda_m(L)-\lambda_2(L)}{\lambda_m(L)+\lambda_2(L)} = \frac{1-\rho}{1+\rho}$ with $\rho = \frac{\lambda_2(L)}{\lambda_m(L)}$ called the condition number of the Laplace matrix L.

The flow characterized by FOS is l_2 minimal [3]. The solution to the l_2 minimization problem "*minimize* $\|x\|_2$ *over all* x *with* $Ax = w^0 - \overline{w}$" is given by $x = A^T z$, where $Lz = w^0 - \overline{w}$ [3],[8]. Now z can be expressed through the eigenvalues λ_i and eigenvectors z_i of L as $z = \sum_{i=2}^m \frac{1}{\lambda_i} z_i$ where $w^0 = \sum_{i=1}^m z_i$ and $w^0 - \overline{w} = \sum_{i=2}^m z_i$. As it has been shown in [3], for a polynomial based load balancing scheme with $w^k = p_k(M)w^0$, the corresponding l_2-minimal flow x^k (such that $w^0 - w^k = Ax^k$) is given by $x^k = q_{k-1}(L)w^0$, the polynomial q_{k-1} being defined by $p_k(1 - \alpha t) = 1 + tq_{k-1}(t)$. The flows x^k converge to x, the l_2-minimal balancing flow. It was also shown in [3] that x^k can be very easily computed along with w^k if the iteration relies on a 3-term recurrence relation (as is the case for FOS, SOS and the OPT scheme of Sect.4).

3 Alternating Direction Iterative Schemes

Cartesian products of lines and paths have been discussed previously in [13], but no analysis of the quality of the flow have been made. We generalize the problem to the product of arbitrary graphs, show that one can reduce the number of loadbalance iterations by alternating the direction of the loadbalancing and analyze the resulting flow. We propose a new Mixed-Direction scheme that results in a better flow than the classical Alternating-Direction scheme.

Let G be the cartesian product of two graphs G' and G'' with vertex sets V' and V'', edge sets E' and E'' and the spectra $\lambda_1(G'), \ldots, \lambda_p(G')$ and $\lambda_1(G''), \ldots, \lambda_q(G'')$, where we denote with $p = |V'|$ and $q = |V''|$. Now G has the vertex set $V = V' \times V''$ and the edge set $E = \{((u'_i, u''_j), (v'_i, v''_j)) \mid$ where $u'_i = v'_i$ and $(u''_j, v''_j) \in E''$ or $u''_j = v''_j$ and $(u'_j, v'_j) \in E'\}$. Any eigenvalue $\lambda(G)$ of G is of the form $\lambda(G) = \lambda_i(G') + \lambda_j(G'')$ with $\lambda_i(G')$ eigenvalue of G' $i \in \{1, \ldots, p\}$ and $\lambda_j(G'')$ eigenvalue of G'' $j \in \{1, \ldots, q\}$. (The same relation also holds for the

eigenvalues of the adjacency matrices.) We denote with I_k the identity matrix of order k. Now the adjacency matrix B_G of G is $B_G = I_q \otimes B_{G'} + B_{G''} \otimes I_p$ where $\otimes$ denotes the tensor product. Obviously, the matrices $I_q \otimes B_{G'}$ and $B_{G''} \otimes I_p$ have a common system of eigenvectors.

Now we apply the *alternating direction iterative* (ADI) load balancing strategy on G, which is similar to the alternating direction method for solving linear systems [11]. In an iteration, firstly balance *among one component* of the cartesian product (for example G') using FOS, and secondly *among the other component* (in our case G'') using FOS. We denote with $M' = 1 - \alpha' L'$ and $M'' = 1 - \alpha'' L''$ the diffusion matrices of G' and G''. The diffusion scheme ADI-FOS then has the form

$$\forall\, e = \{(u'_i, u''_j), (u'_i, v''_l)\} \in E:\ y_e^{k-1} = \alpha''(w^{k-1}_{(u'_i,u''_j)} - w^{k-1}_{(u'_i,v''_l)}); \text{ and} \tag{2}$$

$$\forall\, e = \{(u'_i, u''_j), (v'_l, u''_j)\} \in E:\ y_e^{k} = \alpha'(w^{k}_{(u'_i,u''_j)} - w^{k}_{(v'_l,u''_j)}); \tag{3}$$

Then we have $w^{i+1} = (M'' \otimes I_p)(I_q \otimes M')w^i = (M'' \otimes M')w^i$.

Theorem 1. *Let G be the cartesian product of two connected graphs G' and G''. Denote L, L', L'' the corresponding Laplacians and $M = I - \alpha L, M' = I - \alpha' L', M'' = I - \alpha'' L''$ the diffusion matrices for the FOS with optimal parameters $\alpha, \alpha', \alpha''$. Moreover, let γ_M and $\gamma_{M'' \otimes M'}$ be the diffusion norms. Then $\gamma_{M'' \otimes M'} < \gamma_M$. If G' and G'' are isomorphic, then $\gamma_{M'' M'} < \gamma_M^2$, i.e. FOS combined with the ADI scheme only needs half of the number of iterations in order to guarantee the same upper bound on the error as FOS.*

Proof. Clearly, $\gamma_{M' \otimes M''} = \max\{\gamma_{M'}, \gamma_{M''}\}$ whereas $\gamma_M = 1 - \frac{2\lambda_2(L)}{\lambda_{pq}(L) + \lambda_2(L)}$. Using $\lambda_2(L) = \min\{\lambda_2(L'), \lambda_2(L'')\}$ and $\lambda_{pq}(L) = \lambda_p(L') + \lambda_q(L'')$, one then concludes $\gamma_{M'} \leq \gamma_M$ and $\gamma_{M''} \leq \gamma_M$. If G' and G'' are isomorphic, then $\frac{\lambda_p(L') - \lambda_2(L')}{\lambda_p(L') + \lambda_2(L')} < \left(\frac{\lambda_p(L') + \lambda_p(L') - \lambda_2(L')}{\lambda_p(L') + \lambda_p(L') + \lambda_2(L')}\right)^2$. □

The flow calculated by this scheme is not minimal in the l_2- norm and we study the flow in the following. Denote by $z_{i,j}$ the common eigenvectors of the matrices $I_q \otimes B_{G'}$ and $B_{G''} \otimes I_p$ and by A' and A'' the edge-incidence matrices of the graphs G' and G''. We assume that the graphs G' and G'' are isomorphic with common eigenvalues μ_i. Now let $A_1 = I \otimes A'^T$ and $A_2 = A''^T \otimes I$. Let w_1^k and w_2^k be the load after k iterations after the balancing in one dimension and after balancing in both dimensions. Let F_1^{k+1} and F_2^{k+1} be the flows in iteration $k+1$ among the first and the second dimension with $F_1^{k+1} = \alpha A_1^T w_2^k$ and $F_2^{k+1} = \alpha A_2^T w_1^{k+1}$. For the total flow in these directions we get

$$F_2 = \sum_{k=0}^{\infty} F_2^{k+1} = \alpha \sum_{k=0}^{\infty} A_2^T \Big(\sum_{i,j=1}^{|V'|} \mu_i (\mu_i \mu_j)^k z_{i,j} \Big)$$

$$= \alpha \sum_{i,j=1}^{|V'|} \mu_i \sum_{k=0}^{\infty} (\mu_i \mu_j)^k A_2^T z_{i,j} = \sum_{i,j=1}^{|V'|} \frac{1 - \alpha\lambda_i}{\lambda_i + \lambda_j - \alpha\lambda_i\lambda_j} A_2^T z_{i,j}$$

$$F_1 = \sum_{k=0}^{\infty} F_1^{k+1} = \alpha \sum_{k=0}^{\infty} A_1^T \Big(\sum_{i,j=1}^{|V'|} (\mu_i \mu_j)^k z_{i,j} \Big)$$
$$= \alpha \sum_{i,j=1}^{|V'|} \sum_{k=0}^{\infty} (\mu_i \mu_j)^k A_1^T z_{i,j} = \sum_{i,j=1}^{|V'|} \frac{1}{\lambda_i + \lambda_j - \alpha \lambda_i \lambda_j} A_1^T z_{i,j}.$$

We observe, that the quality of the flow depends on the value of α. The result of the classical diffusion method is a l_2-minimal flow. In this case we get

$$F = \sum_{k=0}^{\infty} \alpha A^T w^k = \alpha \sum_{l=1}^{|V|} \sum_{k=0}^{\infty} \mu_l^k(M) A^T z'_l = \sum_{i,j=1}^{|V'|} \frac{1}{\lambda_i + \lambda_j} A^T z_{i,j},$$

with z'_l being the eigenvectors of the diffusion matrix M to the eigenvalues $\mu_l(M)$ which are identical to the eigenvectors $z_{i,j}$ of the Laplacian L with the eigenvalues $\lambda_i + \lambda_j$. We observe that the l_2-minimal flow does not depend on α whereby the flow of the ADI-method will only converge to the l_2-minimal flow if α converges to zero. This implies a larger number of iterations.

To give an example, the $n \times n$ torus with an initial load of $w^0 = Kz_{1,1} + Kz_{n,n}$, for which the optimal value for α converges to $\frac{1}{2}$ with an increasing n. Therefore, the flows F_1 and F_2 for each edge are also increasing with increasing n. It turns out that with this scheme and this initial load, the load is only slightly balanced in each iteration, whereas a large amount of load is added on all circles of length 4. The result is a final flow with many heavy-weighted circles.

Therefore, we construct a new *mixed direction iterative* (MDI) scheme. In each iteration with an even number we first balance *among component 1* and then *among component 2*, whereas in each iteration with an odd number e first balance *among component 2* and then *among component 1*, Thus, the order of the components for each sub-iteration is changed for each iteration. With similar arguments as before, the diffusion norm is the same as for ADI, but we get the flows

$$F_1 = \sum_{i,j=1}^{|V'|} \frac{1 + \mu_i \mu_j^2}{1 - \mu_i^2 \mu_j^2} A_1^T z_{i,j} \quad \text{and} \quad F_2 = \sum_{i,j=1}^{|V'|} \frac{\mu_i + \mu_i \mu_j}{1 - \mu_i^2 \mu_j^2} A_2^T z_{i,j}.$$

Now, for the same load w^0 as before ($w^0 = Fz_{1,1,} + Fz_{n,n}$), we can observe that, if n increases, the flows F_1 and F_2 will be bounded. Although the flow is not necessarily bounded for other initial load distributions, MDI generally calculates a smaller flow than ADI. This fact can also be seen in the experiments of Sect. 5.

4 Optimal Schemes

An optimal scheme OPS has been introduced in [3]. We present another optimal scheme OPT with the same properties but with a much simpler construction. As it was shown in Sect. 2, the error e^k of any polynomial based scheme satisfies $e^k = p_k(M)(\sum_{i=2}^m z_i) = \sum_{i=2}^m p_k(M) z_i = \sum_{i=2}^m p_k(\mu_i) z_i$. Now we apply a variant of the local iterative algorithm (1) where the parameter α varies from step to

step. More precisely, given an arbitrary numbering λ_k, $1 \leq k \leq m-1$, for the non-zero eigenvalues of L, we take $\alpha = \alpha_k = \frac{1}{\lambda_k}$ to get

$$\forall\, e = \{v_i, v_j\} \in E:\ y_e^{k-1} = \frac{1}{\lambda_k}(w_i^{k-1} - w_j^{k-1}); \quad x_e^k = x_e^{k-1} + y_e^{k-1};$$
$$\text{and } w_i^k = w_i^{k-1} - \sum\nolimits_{e=\{v_i,v_j\}\in E} y_e^{k-1} \tag{4}$$

In each iteration k, a node i adds a flow of $\frac{1}{\lambda_k}(w_i^{k-1} - w_j^{k-1})$ to the flow over edge $\{i,j\}$, choosing a different eigenvalue for each iteration. We obtain

$$w^k = \left(I - \frac{1}{\lambda_{k+1}}L\right) w^{k-1} \qquad \text{and} \qquad e^{m-1} = \prod_{i=2}^{m}\left(I - \frac{1}{\lambda_i}L\right)\sum_{j=2}^{m} z_j = 0,$$

for the weight w^k after k iterations and the final error e^{m-1}. It shows that the load is perfectly balanced after $m-1$ iterations and the flow is l_2-minimal as stated in Sect. 2. For each iteration a different eigenvalue of L is used, eliminating one eigenvector at a time. The order of the eigenvalues may be arbitrary, however, it may influence the numerical stability of the calculation. Even with a favorable order, the new scheme will generally be more sensitive to rounding errors than the OPS scheme from [3]. This issue will be discussed in Sect. 5. No further parameters besides the eigenvalues have to be used for this scheme and it is a much simpler optimal scheme than the one in [3].

Consider for example the k-dimensional hypercube with an initial load of $w^0 = (K, 0, \ldots, 0)$. The dimension exchange strategy is well-known for load balancing on the hypercube, where in each iteration only the balancing in one dimension is allowed and a node equalizes its load with the load of its neighbor. Thus, error e^k decreases to zero after k steps and the l_2-norm of the resulting flow is $F_{de} = K\sqrt{\frac{2^k-1}{2^{k+1}}}$. The k-dim. hypercube has $k+1$ distinct eigenvalues $0, 2, 4, \ldots, 2k$ and in the algorithm of the OPT scheme the new flow y_e^t over edge $e = \{v_i, v_j\}$ in iteration t is simply $y_e^t = \frac{1}{2t}(w_i^t - w_j^t)$. OPT calculates the l_2-minimal flow $F_{l_2} = \sqrt{\sum_{i=0}^{d-1}\left(\frac{K - \frac{K}{2^k}(\sum_{j=0}^{i}\binom{d}{j})}{\binom{d}{i}(d-i)}\right)^2}$, much smaller than F_{de}.

The main disadvantage of the OPT scheme is the fact that it requires all eigenvalues of the graph. Although their calculation can be very time-consuming for large graphs, they can easily be computed for small processor networks and they are known for many classes of graphs like e.g. paths, cycles, stars, hypercubes, grids or tori, which often occur as processor networks.

In Sect. 3 we have introduced ADI-FOS, now ADI-OPT is introduced.

Theorem 2. *Let G be a graph and $G \times G$ the cartesian product of it. The applying of the OPT scheme (4) for each direction of ADI gives a loadbalancing scheme ADI-OPT which only needs $m-1$ iterations for the loadbalancing on $G \times G$, where m is the number of distinct eigenvalues of the Laplacian of G.*

Proof. Applying OPT for $G \times G$ with $G = (V, E)$ we get $w^i = (I - \frac{1}{\lambda_i} L')w^{i-1}$, $1 \leq i \leq \frac{m(m+1)}{2} - 1$ and λ_i is a nonzero eigenvalue of $L' = I \otimes L + L \otimes I$. By combining OPT with ADI we obtain $w^i = (I - \frac{1}{\lambda_{i+1}} I \otimes L)(I - \frac{1}{\lambda_{i+1}} L \otimes I)w^{i-1}$, where λ_{i+1} is a nonzero eigenvalue of $I \otimes L$ and $L \otimes I$. Then

$$e^{m-1} = \prod_{i=2}^{m} (I - \frac{1}{\lambda_i} I \otimes L)(I - \frac{1}{\lambda_i} L \otimes I)e^0 = \prod_{i=2}^{m} (I - \frac{1}{\lambda_i} I \otimes L)(I - \frac{1}{\lambda_i} L \otimes I) \sum_{k,j \neq 1} z_{k,j} = 0,$$

where $z_{i,j}$ are the corresponding eigenvectors of $L' = I \otimes L + L \otimes I$. □

The Laplacian matrix of $G \times G$ can have up to $\frac{m(m+1)}{2}$ distinct eigenvalues, whereas the new ADI-OPT scheme uses only $m - 1$ distinct eigenvalues of the Laplacian of G to eliminate all eigenvectors and to reduce the error to zero.

5 Experiments

Firstly, we perform experiments to find out in which order the $m - 1$ distinct non-zero eigenvalues $\lambda_2 < \lambda_3 < ... < \lambda_m$ in the OPT scheme are to be used in the iterations. Although the load after $m - 1$ iterations is, in theory, independent of the afore chosen order, one may trap in numerical instable conditions with some orders. The load balancing is performed on a path of 32 vertices ($m = 32$), for which we observed the most difficult numerical problems, and the results are presented in Fig. 1. It is stopped after $m - 1 = 31$ iterations. In the initial load distribution RAN, $100 * |V|$ load elements are randomly distributed among the $|V|$ nodes (different random distributions exhibited the same behavior). With an increasing order of eigenvalues (λ_2,λ_3,λ_4,...), the error be-

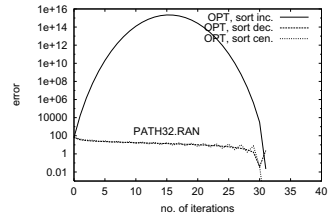

Fig. 1. Optimal OPT scheme on a path of 32 nodes with sorting of the eigenvalues in increasing, decreasing and center-started order. The initial load distribution is random.

comes very high after a few iterations and the final error is about 0.01, due to numerical problems with the high numbers. With a decreasing order (λ_m, λ_{m-1}, λ_{m-2}, ...), the error decreases monotonously until the last iteration, where it increases to a final error of about 1. The best behavior can be observed

with the center-started order ($\lambda_{\frac{m}{2}}$,$\lambda_{\frac{m}{2}-1}$,$\lambda_{\frac{m}{2}+1}$,$\lambda_{\frac{m}{2}-2}$,$\lambda_{\frac{m}{2}+2}$,... for even m and $\lambda_{\frac{m-1}{2}}$,$\lambda_{\frac{m-1}{2}+1}$,$\lambda_{\frac{m-1}{2}-1}$,$\lambda_{\frac{m-1}{2}+2}$,$\lambda_{\frac{m-1}{2}-2}$,... for odd m). Although the error alternately increases/decreases due to the changing order of high and low eigenvalues, the final error is almost zero and we use this very robust order for the following experiments. An even better order can be the use of Leja Points [9].

We compare the new optimal scheme OPT with the First-Order scheme (FOS) and the optimal scheme OPS in [3] on a hypercube of dimension 6 ($m = 7$) and an 8×8 torus ($m = 13$) in Fig. 2. Apart from the RAN initial distribution we use PEAK, in which one node has a load of $100 * |V|$ and the others 0. The balancing stops after t iterations when the error $||w^t - \overline{w}||_2$ is less than 10^{-6}. Both optimal schemes behave similarly to FOS for the first iterations, but then they suddenly drop down to 0 after $m - 1$ iterations. OPS and OPT show the same behavior, but our new optimal scheme OPT has a much simpler construction.

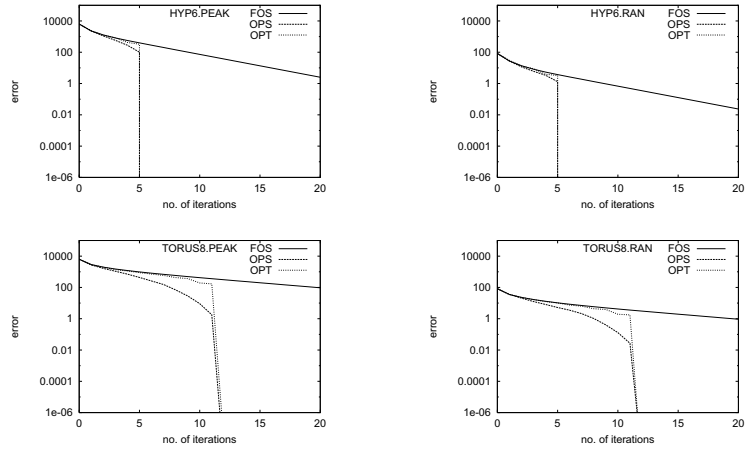

Fig. 2. Iterative Diffusion Loadbalancing on a hypercube of dimension 6 with initial PEAK (upper left) and RAN (upper right) load distribution and on an 8×8 Torus with initial PEAK (lower left) and RAN (lower right) load distribution.

The 16×16 torus is the cartesian product of a cycle with 16 vertices. Tab. 1 shows how the flow depends on α for the ADI-FOS scheme with initial load PEAK. It confirms the theoretical analyses that with decreasing values of α the flow tends to the l_2-minimal flow, but also the number of iterations increases.

Tab. 2 compares the FOS and OPT schemes with the ADI and MDI modifications. Both the FOS and OPT schemes compute the same l_2-minimal flow (Sect. 2), whereas the flow in the ADI and MDI case may differ from each other. As proven before, neither of their flows is minimal in the l_2 norm and they are also worse with respect to the l_1 and l_∞ norms of our test cases. Furthermore, the flow of ADI-OPT is much smaller than of ADI-FOS in all norms and it is

Table 1. Flow for ADI-FOS scheme with different values for α on a 16×16 torus.

	$\alpha = 0.49$	$\alpha_{opt} = 0.4817$	$\alpha = 0.4$	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.01$	l_2-min. flow
l_1	627200	355082.51	207242.41	204800	204800	204800	204800
l_2	52500.69	39311.89	21361.38	18188.09	17967.10	17919	17918.62
l_∞	19066.10	16743.38	10828.53	7637.55	6908.24	6422.10	6375
iter.	528	291	349	708	1427	14366	

only a small fraction larger than the l_2-minimal flow. The flow of MDI improves over ADI in all test cases. The number of iterations for ADI and MDI are the same and are always smaller than the examples without ADI or MDI. As proven before, the upper bound on the number of iterations for FOS is twice as high as for ADI-FOS. In the experiments ADI-FOS halves the number of iterations for the RAN case and almost halves them for the PEAK case. For OPT, the number of iterations is $m - 1$ with $m = 41$ for the 16×16 torus and $m = 9$ for the cycle of 16 vertices (the product being the 16×16 torus). Thus, the number of iterations will reduce from 40 to 8 if the ADI or MDI schemes are used.

Table 2. Flow of ADI/MDI for PEAK(left) and RAN(right) loads on a 16×16 torus.

			ADI-		MDI-				ADI-		MDI-	
	FOS	OPT	FOS	OPT	FOS	OPT	FOS	OPT	FOS	OPT	FOS	OPT
l_1	204800	204800	355083	204800	205663	204800	2581	2581	4123	2727	3048	2716
l_2	17919	17919	39312	20235	23700	19993	142	142	228	152	171	152
l_∞	6375	6375	16743	9927	12185	9751	19.4	19.4	29.9	19.7	22.3	19.7
iter.	578	40	291	8	291	8	458	40	229	8	229	8

Acknowledgment. We thank Holger Arndt from Wuppertal for his suggestions improving an earlier version of this manuscript.

References

[1] D.M. Cvetkovic, M. Doob, and H. Sachs. *Spectra of Graphs*. Johann Ambrosius Barth, 3rd edition, 1995.

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

[3] R. Diekmann, A. Frommer, and B. Monien. Efficient schemes for nearest neighbor load balancing. In G. Bilardi et al., editor, *European Symposium an Algorithms*, LNCS 1461, pages 429–440. Springer, 1998.

[4] R. Diekmann, D. Meyer, and B. Monien. Parallel decomposition of unstructured FEM-meshes. *Concurrency: Practice and Experience*, 10(1):53–72, 1998.

[5] R. Diekmann, F. Schlimbach, and C. Walshaw. Quality balancing for parallel adaptive FEM. In *IRREGULAR'98*, LNCS 1457, pages 170–181. Springer, 1998.

[6] B. Ghosh, S. Muthukrishnan, and M.H. Schultz. First and second order diffusive methods for rapid, coarse, distributed load balancing. In *SPAA*, pages 72–81, 1996.
[7] Y.F. Hu and R.J. Blake. An improved diffusion algorithm for dynamic load balancing. *Parallel Computing*, 25:417–444, 1999.
[8] Y.F. Hu, R.J. Blake, and D.R. Emerson. An optimal migration algorithm for dynamic load balancing. *Concurrency: Prac. and Exp.*, 10(6):467–483, 1998.
[9] L. Reichel. The application of leja points to richardson iteration and polynomial preconditioning. *Linear Algebra and its Applications*, 154-156:389–414, 1991.
[10] K. Schloegel, G. Karypis, and V. Kumar. Parallel multilevel diffusion schemes for repartitioning of adaptive meshes. In *Proc. EuroPar'97*, LNCS. Springer, 1997.
[11] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, 1962.
[12] C. Walshaw, M. Cross, and M. Everett. Dynamic load-balancing for parallel adaptive unstructured meshes. In *SIAM Conf. on Parallel Proc. for Sci. Comp.*, 1997.
[13] C. Xu and F.C.M. Lau. *Load Balancing in Parallel Computers*. Kluwer, 1997.