Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

J. Parallel Distrib. Comput. 64 (2004) 481–497

Journal of Parallel and Distributed Computing

http://www.elsevier.com/locate/jpdc

Dynamic load balancing by diffusion in heterogeneous systems

Tiberiu Rotaru[a,]* and Hans-Heinrich Nägeli[b]

[a] Fraunhofer-Institut für Techno- und Wirtschaftsmathematik, Competence Center "High Performance Computing" (CC HPC), Gottlieb-Daimler-Straße, Gebaude 49 DE-67663 Kaiserslautern, Germany

[b] Institut d'informatique, Université de Neuchâtel, Émile-Argand 11, CH-2007 Neuchâtel, Switzerland

Received 15 December 2000; revised 22 April 2003

Abstract

The distributed environments constitute a major option for the future development of high-performance computing. In order to be able to efficiently execute parallel applications on such systems, one should ensure a fair utilization of the available resources. Here, we address a number of aspects regarding the generalization of the diffusion algorithms for the case when the processors have different relative speeds and the communication parameters have different values. Although some work has been done in this direction, we propose complementary results and we investigate other variants than those commonly used. In a first step, we discuss general aspects of the generalized diffusion. Bounds are formulated for the convergence factor and an explicit expression is given for the migration flow generated by such algorithms. It is shown that this flow has an important property, that is a scaled projection of all other balancing flows. In the second part, a variant of generalized diffusion is investigated. Complexity results are formulated and it is shown that this algorithm theoretically converges faster than the hydrodynamic algorithm. Comparative tests between different variants of generalized diffusion algorithms are performed.
© 2004 Published by Elsevier Inc.

Keywords: Heterogeneous environments; Dynamic load balancing; Generalized diffusion; Balancing flow; Adaptive computations

1. Introduction

A main concern when using distributed computing environments for running parallel applications is the efficient use of the available computational resources. As the execution time of the entire application is determined by the slowest processor, the total workload must be fairly distributed over processors, i.e. proportionally to their speeds. From time to time, it may be necessary to proceed to a redistribution of the workloads, as during the execution both the workload affected to a processor and its processing capacity may change, often in an unpredictable manner. Parallel adaptive finite element applications arising in computational fluid dynamics or computational mechanics constitute typical applications that require adequate dynamic load balancing techniques [11,15]. Such applications rely on unstructured meshes, in which the nodes represent finite elements/volumes or vertices of finite elements/volumes and the links between them reflect neighborhood relations. During the computations, the adaptive meshes may undergo significant changes due to refinement/de-refinement operations, resulting in an uneven partitioning. It is well known that in distributed systems the inter-processor communication is costly; therefore, a fair workload re-distribution should be done with a minimum cost of data migration and should minimize the number of dependencies between nodes residing on different processors. A centralized approach leads to heavy communication and it is not scalable. A dynamic load balancing algorithm should use local communication and should be able to rapidly compute a new fair data distribution. The nearest-neighbor algorithms meet these requirements to a large extent. For this reason they were intensively studied in the last years, mainly in the context of homogeneous systems. They lead an unbalanced system to a global "equilibrium" state by exchanging information/workload only between processors owning neighboring subdomains. The most popular variants are the diffusion [1,4,6,12,14], the dimension exchange [4,19,30], the multi-level diffusion [13] and the gradient model [29].

*Corresponding author. Fax: +49-631-303-1833.
E-mail addresses: rotaru@itwm.fhg.de (T. Rotaru), hans.naegeli@unine.ch (H.-H. Nägeli).

Less attention has been paid to dynamic load balancing in heterogeneous systems, but some important steps have been already done in this direction. Hui and Chanson [17,18] proposed an intuitive approach based on a hydrodynamic analogy, for a heterogeneous environment characterized by different computing powers and uniform communication. Watts and Taylor [28] proposed for a similar heterogenous environment implicit diffusion schemes. Diekmann et al. [5] proposed diffusion schemes for a computational environment characterized by uniform computing powers and different communication parameters. Elsässer et al. [8] extended these schemes for computational environments that are heterogeneous both with respect to the processing performances and the communication speeds. A generalization of the diffusion algorithm for the case when the processors have different speeds and the communication is uniform was discussed in [24]. Here, we discuss some general aspects regarding the heterogeneous diffusion algorithms and we investigate a particular variant of diffusion that appears as a generalization of the algorithm proposed by Boillat in the homogeneous case [1]. A generalization of the algorithm proposed by Cybenko was investigated in [7,8]. Our contribution can be summarized as follows: we give a direct explicit expression of the balancing flow generated by a generalized diffusion algorithm and we show that this flow has an interesting property, that it is a scaled projection of any other balancing flow in the same heterogeneous environment. We give estimations for the second largest eigenvalue of a generalized diffusion matrix and we estimate the complexity of the proposed algorithm. We further show that this algorithm has a better convergence factor than the hydrodynamic algorithm [17,18]. Compared to other approaches, the one we consider here offers the advantage of not using parameters that are dependent upon the eigenvalues of the Laplacian of the communication graph.

2. General results related to diffusion

In the sequel, we assume in our model that the processors may have different relative speeds and we abstract the speed of a processor by a real positive number. A processor's workload is considered to be infinitely divisible, so it can be represented through a real positive number, too. Different communication speeds are also assumed. It is considered that the communication topology may change during the computations, between successive load redistribution phases.

We consider a heterogeneous computing model (G, l, c, w) in which:

- $G = (V, E)$ is a connected graph whose vertices correspond to the processors and whose edges reflect dependencies between data residing on different processors. Let $V = \{1, \dots, p\}$ and $E = \{e_1, e_2, \dots, e_q\}$.
- l is the vector of the processors' workloads;
- c is the vector of the processors' speeds; without loss of generality we shall consider that c is normalized relative to the 1-norm, i.e. $\sum_{1 \leqslant i \leqslant |V|} c_i = 1$.
- w_k, $1 \leqslant k \leqslant q$, are weights associated to the edges of G.

The following notation is also used:

- e is the vector of size p having all entries equal to 1;
- $c^{1/2}$ is the *vector of the square roots of the speeds*;
- $c_{\max}$ and $c_{\min}$ are the *maximum and the minimum speed*;
- For a given vector u, $\operatorname{diag}(u)$ denotes the diagonal matrix whose elements on the main diagonal are the elements of u. We note $D = \operatorname{diag}(c)$ and $W = \operatorname{diag}(w)$;
- $\bar{l}$ is the workload vector corresponding to the *fair distribution*, i.e. $\bar{l}_i / c_i = \bar{l}_j / c_j$, for all i and j;
- δ_i is the *degree of the vertex i* and $\delta_i^w = \sum_{i \in e_k} w_k$ is the *weighted degree* of i;
- $\Delta = \max_{i=1,p} \delta_i$ is *the maximum degree* of G, $\Delta^w = \max_{i=1,p} \delta_i^w$ is the *maximum weighted degree* of G;
- $e(G)$ is the *edge connectivity* of the graph G; it represents the minimum cardinal number of a set of edges whose removal disconnects the graph;
- $\mathcal{N}(i)$ denotes the *set of neighbors* of a the vertex i;
- $\operatorname{diam}(G)$ is the diameter of the graph G;
- With respect to an arbitrary orientation of the edges of G, the vertices that define an arc can be categorized as *head* or *tail*, depending upon their position in the ordered pair. The *vertex-edge incidence matrix A* of G is a $p \times q$ matrix such that

$$a_{ij} = \begin{cases} 1 & \text{if } i \text{ is the head of } e_j, \\ -1 & \text{if } i \text{ is the tail of } e_j, \\ 0 & \text{otherwise.} \end{cases}$$

  One can easily remark that $A^T e = 0$;
- A flow $f \in \mathbb{R}^q$ on the edges of G is called a *balancing flow* if $Af = l - \bar{l}$;
- $\|\cdot\|_q$ denotes the q-norm, defined as $\|x\|_q = (\sum_{i=1}^p |x_i|^q)^{1/q}$ $\forall x \in \mathbb{R}^p$;
- $\|\cdot\|_{q,W}$ denotes the weighted q-norm, defined as $\|x\|_{q,W} = \|Wx\|_q$, for any vector $x \in \mathbb{R}^p$ and W a diagonal matrix with positive elements;
- For any square matrix M, $\rho(M)$ denotes its *spectral radius* and λ_M *the second largest eigenvalue in absolute value of M*.

The efficient execution of parallel applications on heterogeneous systems requires adequate load redistribution techniques. Such methods should be fast, scalable and should minimize the communication overhead. We consider the following problem in a heterogeneous computing environment: what is the amount of workload to shift between adjacent vertices in the communication graph induced by the application (that is mapped onto a heterogeneous network of processors), so that each processor has the same proportion of workload to process relative to its speed and the cost of data migration is minimized. The minimum discussed here is with respect to the norm $||\cdot||_{2,W^{-1/2}}$, as the diffusion schemes naturally minimize it [5,8,14].

2.1. Generalized diffusion

The generalized diffusion algorithm (shortly, GDA) is a generalization of the classical diffusion algorithm and it reads as in Algorithm 1.

Algorithm 1. Generalized diffusion algorithm (GDA)

```
$k = 0$;
while (not converged) do
    for all $i$ do
        for all neighbors $j$ of $i$ do
            send $l_i^{(k)}$ to $j$;
            receive $l_j^{(k)}$ from $j$;
            $\delta_{ij}^{(k)} = m_{ji}\, l_i^{(k)} - m_{ij} l_j^{(k)}$;
            $l_i^{(k+1)} = l_i^{(k)} - \sum_{\{i,j\}\in E(G)} \delta_{ij}^{(k)}$;
            $k = k + 1$;
        end for
    end for
end while
```

Here, $l^{(0)}$ is the initial workload vector, $l^{(n)}$ is the workload vector after the nth iteration and the m_{ij}s are diffusion parameters. The algorithm can be expressed in a matrix form as an iterative process of type $l^{(n+1)} = Ml^{(n)}$, with M a $p \times p$ nonnegative matrix such that

$$m_{ij} > 0 \quad \text{iff } \{i,j\} \in E \text{ or } i = j,$$
$$\sum_{i\in V} m_{ij} = 1 \quad \text{for all } j \in V,$$
$$m_{ij}c_j = m_{ji}c_i \quad \text{for all } i,j \in V. \tag{2.1}$$

In a computational environment as that assumed here, a matrix that satisfies the above properties is referred to as a *generalized diffusion matrix* (GDM). The first condition means that at each iteration step, the communication is allowed only between neighbors. The second constraint guarantees the workload conservation in the absence of external disturbances. The third condition ensures that when the system has reached the fair state, no load is migrated anymore between neighbors. In the theory of reversible Markov chains the last equations are also referred to as *the balance equations*.

2.2. Convergence and metrics

In a heterogeneous model as that assumed here, a matrix that satisfies the set of conditions (2.1) is primitive and has the spectral radius 1. The unique eigenvector corresponding to the spectral radius, normalized w.r.t. the norm $||\cdot||_1$, is referred to as the *Perron vector* [21]. It is easy to see the *Perron vector* of any GDM coincides with the vector c of the processors' speeds.

According to the Perron–Frobenius theorem [21,27], c and its multiples are the only eigenvectors of M with all entries nonnegative. The fair workload vector is the Perron vector multiplied by the sum of the workloads, i.e. $\bar{l} = c(e^T l^{(0)})$. An important property of the matrices that satisfy the set of equations (2.1) is the following.

Theorem 1. *Any GDM is diagonalizable and all its eigenvalues are real.*

Proof. A GDM M is similar to $D^{-1/2}MD^{1/2}$, which is real symmetric. Therefore, the eigenvalues of M are real. $\square$

We assume that the eigenvalues of M are indexed so that

$$1 = \lambda_1 > \lambda_2 \geqslant \cdots \geqslant \lambda_p > -1.$$

For a generic GDM M, λ_M will denote the *second largest eigenvalue* in absolute value of M, i.e. $\lambda_M = \max\{|\lambda_2|, |\lambda_p|\}$. To be consistent with the homogeneous case [1], the term *convergence factor* is used when referring to it.

The load imbalance can be characterized using various metrics. Thus, for any sequence of workload vectors $\{l^{(n)}\}_{n\geqslant 0}$, obtained by repeated application of a diffusion operator, one can prove the following inequalities:

$$||l^{(n+1)} - \bar{l}||_{2,D^{-1/2}} \leqslant \lambda_M ||l^{(n)} - \bar{l}||_{2,D^{-1/2}},$$
$$||l^{(n)} - \bar{l}||_{2,D^{-1/2}} \leqslant \lambda_M^n ||l^{(0)} - \bar{l}||_{2,D^{-1/2}},$$

$$||l^{(n+1)} - \bar{l}||_{\infty,D^{-1}} \leqslant ||l^{(n)} - \bar{l}||_{\infty,D^{-1}},$$
$$||l^{(n)} - \bar{l}||_{\infty,D^{-1}} \leqslant \frac{\lambda_M^n}{c_{\min}} ||l^{(0)} - \bar{l}||_{\infty,D^{-1}},$$

$$||l^{(n+1)} - \bar{l}||_1 \leqslant ||l^{(n)} - \bar{l}||_1,$$
$$||l^{(n)} - \bar{l}||_1 \leqslant \frac{1}{c_{\min}} \lambda_M^n ||l^{(0)} - \bar{l}||_1.$$

These relations show that after any iteration step, the distance to the fair distribution induced by the weighted 2-norm decreases. Also, the distances induced by the weighted ∞-norm and the 1-norm, eventually decrease. The algorithm results in a geometrical reduction of the above quantities. The convergence rate strongly depends on λ_M, whatever metric is used. The above results show that any GDA completes in $O(1/|\ln(\lambda_M)|)$ steps.

2.3. Theoretical bounds for the convergence factor

Before deciding to proceed to a new workload redistribution, one should dispose of some kind of estimation of the cost of application of a dynamic load balancing algorithm. Here we try to answer to the question "*what is the maximum number of steps that a GDA will take to balance the system?*" Clearly, faster polynomial schemes can be derived in relation with a GDM as described in [5,8,14,23]. However, often such schemes use parameters that are dependent on the eigenvalues of the Laplacian of the communication graph. In a dynamic context in which the communication graph induced by the application changes between successive applications of the load balancing algorithm, these eigenvalues should be recomputed each time such changes occur, which means a supplementary overhead. As the communication graph is often irregular, these eigenvalues cannot be given in advance. Therefore, in order to be used in dynamic contexts, such algorithms should be coupled with fast routines for the computation of the eigenvalues. We consider here the case of *first-order schemes*; polynomial schemes can be easily derived as shown in [8].

We are concerned with finding diffusion matrices with a small convergence factor in heterogeneous computing environments. To achieve this goal, adequate estimations of the convergence factor as a function of the communication graph's characteristics should exist. Bounds can be formulated by analogy with the discrete time reversible Markov chains, as the transposed of a GDM defines a random walk on the weighted graph G. In theory, greater importance is given generally to the second largest eigenvalue rather than to the smallest one. This is motivated by that one may consider $M' = (\frac{1}{2})(I+M)$ instead of M (in this case $\lambda_M{}' = \lambda_2(M')$), without a significant impact on the convergence rate [20,25].

If $u_1, \ldots, u_q$ are arbitrary weights on the edges of G and $U = \mathrm{diag}(u)$, the matrix $L = AUA^T$ is a *weighted Laplacian matrix* of G. The *generalized Laplacian* matrix of G is expressed as $\mathscr{L} = D^{-1/2}LD^{-1/2}$. An important property of these matrices is that they have only nonnegative eigenvalues, with the smallest eigenvalue 0. The vectors e^T and e are left and right eigenvectors of L corresponding to the eigenvalue 0. In the same time, the vectors $c^{T^{1/2}}$ and $c^{1/2}$ are left and right eigenvectors of $\mathscr{L}$ corresponding to the eigenvalue 0.

Conforming to the Courant–Fischer principle, after some simple transformations, the following characterizations can be given for the second smallest eigenvalues of L and $\mathscr{L}$:

$$\mu_2(L) = \min_{\substack{y\neq 0\\ y^Te=0}} \frac{\sum_{e_k=\{i,j\}\in E} u_k(y_i-y_j)^2}{\sum_{i\in V} y_i^2}, \tag{2.2}$$

$$\mu_2(\mathscr{L}) = \min_{\substack{y\neq 0\\ y^Tc=0}} \frac{\sum_{e_k=\{i,j\}\in E} u_k(y_i-y_j)^2}{\sum_{i\in V} c_i y_i^2}. \tag{2.3}$$

Because G is connected, one has $\mu_2(L)>0$ [10].

One can use the isoperimetric constant [22] (w.r.t. the generalized Laplacian) for bounding $\mu_2(\mathscr{L})$. This is defined as follows:

$$i(G) = \min\left\{ \frac{\sum_{i\in S}\sum_{j\notin S} s_{ij}}{\sum_{i\in S} c_i} \middle| S\subset V, 0<\sum_{i\in S} c_i \leqslant \frac{1}{2} \right\}. \tag{2.4}$$

There is a close connection between the spectra of the Laplacian matrices of a graph and that of any GDM on this graph. Indeed, let M be an arbitrary GDM and $L = D - MD$. As the set of conditions (2.1) ensures that $m_{ij}>0$ for all $\{i,j\}\in E(G)$, L can be put in the form $L = AUA^T$, with $u_k = m_{ij}c_j$, for all $e_k = \{i,j\}\in E$. Then, M can be expressed as follows:

$$M = I - D^{1/2}\mathscr{L}D^{-1/2}. \tag{2.5}$$

In this case, the isoperimetric constant coincides with the Cheeger constant of the weighted graph G [3] and with the conductance of the Markov chain defined by M^T [25]. Sinclair and Jerrum [26] showed that this satisfies the Cheeger inequality

$$\frac{i^2(G)}{2} \leqslant \mu_2(\mathscr{L}) = 1 - \lambda_2(M) \leqslant 2i(G). \tag{2.6}$$

The bounds for the second smallest eigenvalue obtained via the Cheeger-type inequalities like (2.6) are often tight [2]. However, it is NP-hard to determine the Cheeger constants [20,22]. We shall prefer to use alternative results.

Theorem 2. *Let M be an arbitrary GDM and $L, \mathscr{L}$ the two Laplacian matrices defined as above. If $\mu_2(L), \mu_2(\mathscr{L})$ are their corresponding smallest nonzero eigenvalues and $c_{\max}$ is the maximum speed, then*

$$\mu_2(L) \leqslant c_{\max}\mu_2(\mathscr{L}).$$

Proof. The proof relies on two alternative characterizations of the second smallest eigenvalues of L and $\mathscr{L}$ that can be obtained using the Lagrange identity from (2.2) and (2.3):

$$\mu_2(L) = \min_{y\neq 0} \max_t \frac{\sum_{\{i,j\}\in E} m_{ij}c_j(y_i-y_j)^2}{\sum_{i\in V}(y_i-t)^2},$$

$$\mu_2(\mathscr{L}) = \min_{y\neq 0} \max_t \frac{\sum_{\{i,j\}\in E} m_{ij}c_j(y_i-y_j)^2}{\sum_{i\in V}c_i(y_i-t)^2}. \qquad \square$$

Fiedler proved in [9] that for any doubly stochastic matrix P, the following inequality holds:

$$\lambda_2(P) \leqslant 1 - 4\sigma_P \sin^2\left(\frac{\pi}{2p}\right), \tag{2.7}$$

where σ_P is the *measure of irreducibility* of P, i.e. $\sigma_P = \min_{\emptyset \neq S \subset V} \sum_{\substack{i \in S \\ j \notin S}} p_{ij}$. Below, we adapt this result for nonsymmetric stochastic matrices.

Theorem 3. *Let M be a GDM. Then*

$$\lambda_2(M) \leqslant 1 - \frac{4}{c_{\max}} \left(\min_{\emptyset \neq S \subset V} \sum_{\substack{i \in S \\ j \notin S}} m_{ij} c_j \right) \sin^2\left(\frac{\pi}{2p}\right). \tag{2.8}$$

Proof. Let $M' = I - L = I - D + MD$. Then M' is a doubly stochastic matrix and the following array of inequalities follows directly from Theorem 2:

$$1 - \lambda_2(M') = \mu_2(L) \leqslant c_{\max}\mu_2(\mathscr{L}) = c_{\max}(1 - \lambda_2(M)).$$

Applying now (2.7) for M', one gets

$$\lambda_2(M) \leqslant 1 - \frac{4}{c_{\max}} \sigma_{M'} \sin^2\left(\frac{\pi}{2p}\right).$$

One should note that $\sigma_{M'} > 0$, as G is connected. □

Corollary 4. *Let M be a GDM and $\theta(M) = \min_{\{i,j\} \in E(G)}\{m_{ij}c_j\}$. Then*

$$\lambda_2(M) \leqslant 1 - 4\frac{e(G)\theta(M)}{c_{\max}} \sin^2\left(\frac{\pi}{2p}\right). \tag{2.9}$$

Proof. The proof is straightforward, by the above theorem and the definition of the edge connectivity. □

Additionally, we prove another useful bound:

Theorem 5. *Let M be an arbitrary GDM, $\theta(M) = \min_{\{i,j\} \in E(G)}\{m_{ij}c_j\}$ and $\operatorname{diam}(G)$ the diameter of the communication graph G. Then*

$$\lambda_2(M) \leqslant 1 - \frac{\theta(M)}{\operatorname{diam}(G)}. \tag{2.10}$$

Proof. Let $\mathscr{L} = I - D^{-1/2}MD^{1/2}$. Following the characterization given in equation (2.3), let z be a vector such that $\sum_i c_i z_i = 0$ and

$$\mu_2(\mathscr{L}) = \frac{\sum_{\{i,j\} \in E(G)} (z_i - z_j)^2 m_{ij} c_j}{\sum_{i \in V(G)} c_i\, z_i^2}.$$

Let i_0 such that $|z_{i_0}| = \max_{i \in V(G)} |z_i| > 0$. As $\sum_i c_i z_i = 0$, there is an index j_0 such that z_{i_0} and z_{j_0} have opposite signs. Let P be a path with minimum length, joining the vertices i_0 and j_0 in the graph G and l its length. It follows that

$$\begin{aligned}
\mu_2(\mathscr{L}) &\geqslant \frac{\sum_{\{i,j\} \in E(P)} (z_i - z_j)^2 m_{ij} c_j}{z_{i_0}^2} \\
&\geqslant \theta(M) \frac{\sum_{\{i,j\} \in E(P)} (z_i - z_j)^2}{z_{i_0}^2} \\
&\geqslant \theta(M) \frac{\left(\sum_{\{i,j\} \in E(P)} (z_i - z_j)\right)^2}{l z_{i_0}^2} \\
&= \frac{\theta(M)}{l} \frac{(z_{i_0} - z_{j_0})^2}{z_{i_0}^2} \geqslant \frac{\theta(M)}{\operatorname{diam}(G)}.
\end{aligned}$$

As $\lambda_2(M) = 1 - \mu_2(\mathscr{L})$, we get relation (2.10). □

2.4. The balancing flow generated by an arbitrary GDA

Usually, the flow generated by various nearest-neighbor schemes is characterized and analyzed indirectly, via its properties. A natural question that arise is what is the expression of the migration flow generated by a GDA? In the sequel, we give a direct explicit formula for this flow, as a function of the generalized Laplacian and of the parameters of the assumed theoretical model.

The following result is useful:

Lemma 6. *If M is an arbitrary GDM and $B = M - ce^T$, then*

(1) *$B^n = M^n - ce^T$ for all $n \geqslant 1$;*
(2) *$\rho(B) = \lambda_M$;*
(3) *$(I - B)^{-1}$ exists, with e^T and c as left and right eigenvectors corresponding to the eigenvalue 1.*

Proof. (1) For $n = 1$, by definition, the relation is true. Suppose that we have $B^n = M^n - ce^T$. As M satisfies the set of conditions (2.1), e^T and c are left and right eigenvectors of M corresponding to the unit eigenvalue and we have $B^{n+1} = (M - ce^T)(M^n - ce^T) = M^{n+1} - ce^T$.

(2) Let $\Lambda(A)$ denote the set of eigenvalues of the square matrix A. The proof relies on that $\Lambda(B) = \Lambda(M) \backslash \{1\} \cup \{0\}$.

(3) The matrix $I - B$ has the smallest eigenvalue $1 - \rho(B) = 1 - \lambda_M > 0$. Therefore $I - B$ is nonsingular. The second part relies on that $e^T B = 0$ and $Bc = 0$. □

Let M be an arbitrary GDM. As shown in Eq. (2.5), M can be put in the form $M = I - A\, UA^T D^{-1}$, with u a vector of size q such that $u_k = m_{ij}c_j$, for all $e_k = \{i,j\}$ and $U = \operatorname{diag}(u)$. As indicated in Algorithm 1, at each iteration step n, an amount

$$\delta_{ij}^{(n)} = m_{ij}c_j\left(\frac{l_i^{(n)}}{c_i} - \frac{l_j^{(n)}}{c_j}\right)$$

is transferred across the edge $e_k = \{i,j\}$. The transfer is considered to take place from node i to node j if $\delta_{ij}^{(n)} \geqslant 0$ and in the opposite direction if $\delta_{ij}^{(n)} < 0$. The *migration flow generated at step n* is the vector $x^{(n)} \in \mathbb{R}^q$ with the entries $x_k^{(n)} = \delta_{ij}^{(n)}$, for each edge e_k with the extremities i and j. In a matrix form, it can be expressed as

$$x^{(n)} = UA^TD^{-1}l^{(n)}. \tag{2.11}$$

The *migration flow* generated by a GDA is then $f = \sum_{n\geqslant 0} x^{(n)}$. The following result identifies the balancing flow generated by a GDA.

Lemma 7. *The flow generated by a GDA has the expression*

$$f = UA^TD^{-1}(I-B)^{-1}l^{(0)}.$$

Proof. First, we notice that $UA^TD^{-1}ce^Tl^{(0)} = 0$, because $A^Te = 0$. As $\rho(B) < 1, I - B$ is nonsingular and $(I-B)^{-1} = \sum_{n\geqslant 0} B^n$. Therefore, the migration flow can be expressed as

$$\begin{aligned} f &= \sum_{n\geqslant 0} x^{(n)} = \sum_{n\geqslant 0} UA^TD^{-1}l^{(n)} \\ &= \sum_{n\geqslant 0} UA^TD^{-1}M^nl^{(0)} \\ &= \sum_{n\geqslant 0} UA^TD^{-1}(B^n + ce^T)l^{(0)} \\ &= \sum_{n\geqslant 0} UA^TD^{-1}B^nl^{(0)} \\ &= UA^TD^{-1}\left(\sum_{n\geqslant 0} B^n\right)l^{(0)} \\ &= UA^TD^{-1}(I-B)^{-1}l^{(0)}. \qquad \square \end{aligned}$$

Theorem 8. *The migration flow generated by a GDA is a balancing flow.*

Proof. By Lemma 6, e^T is a left eigenvalue of $(I-B)^{-1}$. Therefore,

$$\begin{aligned} Af &= AUA^TD^{-1}(I - M + ce^T)^{-1}l^{(0)} \\ &= (I-M)(I-M+ce^T)^{-1}l^{(0)} \\ &= l^{(0)} - ce^T(I-B)^{-1}l^{(0)} \\ &= l^{(0)} - ce^Tl^{(0)} = l^{(0)} - \bar{l}. \qquad \square \end{aligned}$$

Lemma 9. *If M is a GDM, u is a q-vector such that $u_k = m_{ij}c_j$, for all $e_k = \{i,j\} \in E(G)$, $U = \operatorname{diag}(u)$ and $B = M - ce^T$, then*

$$P = U^{1/2}A^TD^{-1}(I-B)^{-1}AU^{1/2}$$

is a projection matrix.

Proof. Clearly, $P = P^T$. On the other hand, by Lemma 6, we have

$$\begin{aligned} P^2 &= U^{1/2}A^TD^{-1}(I-B)^{-1}AUA^TD^{-1}(I-B)^{-1}AU^{1/2} \\ &= U^{1/2}A^TD^{-1}(I-B)^{-1}(I-B-ce^T)(I-B)^{-1}AU^{1/2} \\ &= P - U^{1/2}A^TD^{-1}(I-B)^{-1}ce^T(I-B)^{-1}AU^{1/2} \\ &= P. \qquad \square \end{aligned}$$

This allows us to formulate a more important result.

Theorem 10. *Let $\mathscr{F}(H)$ denote the set of all balancing flows in $H = (G,l,c,w)$, f the flow generated by a GDA in H and P the projection matrix defined above. Then*

$$f = U^{1/2}PU^{-1/2}x \ \forall x \in \mathscr{F}(H).$$

Proof. Let x be a balancing flow in H. Replacing l by $Ax + \bar{l}$ in the expression of f, one gets

$$\begin{aligned} f = {} & UA^TD^{-1}(I-B)^{-1}Ax \\ & + UA^TD^{-1}(I-B)^{-1}ce^Tl. \end{aligned}$$

The second term in the right-hand side of the above equation is 0; therefore, one has $f = U^{1/2}PU^{-1/2}x$. $\square$

It is well known that in the homogeneous case the diffusion schemes generate a balancing flow that is minimal in the 2-norm [5,14]. This is also true in the case of a generalized diffusion depending on a single scalar like that proposed in [8]. Here, we proved a more general result, that the flow generated by any GDA is a scaled projection of all balancing flows in the same heterogeneous environment. The minimality with respect to the norm $\|\cdot\|_{2,U^{-1/2}}$ of the flow generated by a GDA follows directly from the above theorem, as $\|P\|_2 = 1$.

3. Analysis of a family of generalized diffusion algorithms

In a heterogeneous context as that assumed here, one is interested in finding "good" generalized diffusion matrices, i.e. GDMs that have a small convergence factor. In the homogeneous case, optimal matrices have been found for the diffusion depending on a single parameter on n-dimensional meshes and n-dimensional tori [30]. In the heterogeneous case, the processors may have different speeds and the communication parameters may have different values, the diffusion matrix is nonsymmetric and finding optimal GDMs becomes a much more difficult task, even for simple cases.

In the homogeneous case, a common practice is to consider diffusion matrices that have all entries corresponding to edges of the communication graph set to a certain α, adequately chosen. For common topologies such as hypercubes, meshes or tori, an easy way to choose this α is to simply set it to $1/(\Delta+1)$ [4,30]. Boillat [1] suggested to take $m_{ij} = 1/(\max\{\delta_i, \delta_j\}+1)$, for all edges $\{i,j\} \in E(G)$. In the heterogeneous case, the GDMs should take into account also the performances of the processors and of the communication. Some work has been done in this direction. In [5,16], uniform speeds and different communication parameters were considered. A variant of generalized diffusion for the case when only the processors have different speeds was described in [24]. In [8], the authors generalized the variant of diffusion proposed by Cybenko [4] for the case when both the processors and the network have different performances.

We consider diffusion matrices of type $M = I - ASWA^TD^{-1}$, with S a diagonal matrix of scalars ($S = \mathrm{diag}(s)$, where s is a vector of scalars of size q). These scalars should be chosen so that M satisfies the conditions expressed in Eq. (2.1). The variant of generalized diffusion investigated by Elssäser et al. [8] relies on a diffusion matrix of this type, when $S = \alpha I$. The authors used $\alpha = 2/(\mu_2(\mathscr{L}) + \mu_p(\mathscr{L}))$, which represents the best choice in the case of a diffusion depending on a single parameter.

In the sequel, we consider the family of GDMs defined as

$$m_{ij}(\varepsilon) = \begin{cases} \dfrac{s_k(\varepsilon)w_k}{c_j} & \text{if } e_k = \{i,j\} \in E, \\ 0 & \text{if } \{i,j\} \notin E, i \neq j, \\ 1 - \sum\limits_{\{k,j\}\in E} m_{kj}(\varepsilon) & \text{if } i = j, \end{cases}$$

where, for any edge $e_k = \{i,j\}$,

$$s_k(\varepsilon) = \min\left\{\frac{c_i}{\delta_i^w + \varepsilon}, \frac{c_j}{\delta_j^w + \varepsilon}\right\}. \tag{3.1}$$

This is clearly a generalization of the diffusion proposed by Boillat [1] in the homogeneous case. Indeed, in the homogeneous case one has $c_i = 1/p$ for all $i = 1,p$ and $w_k = 1$, for all $k = 1,q$. For $\varepsilon = 1$, one gets

$$m_{ij}(1) = \begin{cases} \min\left\{\dfrac{1}{\delta_i + 1}, \dfrac{1}{\delta_j + 1}\right\} & \text{if } \{i,j\} \in E, \\ 0 & \text{if } \{i,j\} \notin E, i \neq j, \\ 1 - \sum\limits_{\{k,j\}\in E} m_{kj}(1) & \text{if } i = j, \end{cases}$$

which is exactly the diffusion considered in the above paper.

For a generalized diffusion algorithm that uses the scalars defined in Eq. (3.1), we have

$$\begin{aligned}\theta(M(\varepsilon)) &= \min_{\{i,j\}\in E(G)} \{m_{ij}(\varepsilon)c_j\} \\ &= \min_{k=1,q} \{s_k(\varepsilon)w_k\}.\end{aligned}$$

Clearly, $\theta(M(\varepsilon)) \geqslant c_{\min} w_{\max}/(\Delta^w + \varepsilon)$. With these notations, using Corollary 4, one gets

$$\lambda_2(M(\varepsilon)) \leqslant 1 - 4e(G)\frac{w_{\min}}{\Delta^w + \varepsilon}\frac{c_{\min}}{c_{\max}}\sin^2\left(\frac{\pi}{2p}\right). \tag{3.2}$$

The above result gives an upper bound for the second largest eigenvalue of $M(\varepsilon)$. On the other hand, one can find an upper bound for the absolute value of the smallest eigenvalue as indicated in the following Lemma.

Lemma 11. *For all $\varepsilon \geqslant 0$, if $\lambda_p(M(\varepsilon)) < 0$, then*

$$|\lambda_p(M(\varepsilon))| \leqslant 1 - 2\frac{\varepsilon}{\Delta^w + \varepsilon}. \tag{3.3}$$

Proof. For simplicity, we write λ_p instead of $\lambda_p(M(\varepsilon))$. It is well known that all the eigenvalues of $M(\varepsilon)$ lie in the union of the Gershgorin disks [27]. Therefore, there is an index i such that

$$|\lambda_p - m(\varepsilon)_{ii}| \leqslant \Lambda_i \tag{3.4}$$

with $\Lambda_i = \sum_{j\neq i} m(\varepsilon)_{ji}$. The matrix $M(\varepsilon)$ being column stochastic and λ_p being negative, the above inequality is equivalent to $|\lambda_p| = -\lambda_p \leqslant 1 - 2m(\varepsilon)_{ii} = -1 + 2(1 - m(\varepsilon)_{ii})$. On the other hand, we have

$$\begin{aligned}1 - m(\varepsilon)_{ii} &= \sum_{k:i\in e_k} s_k(\varepsilon)\frac{w_k}{c_i} \\ &= \sum_{\substack{j\in\mathscr{N}(i)\\ e_k=\{i,j\}}} \min\left\{\frac{c_i}{\delta_i^w + \varepsilon}, \frac{c_j}{\delta_j^w + \varepsilon}\right\}\frac{w_k}{c_i} \\ &\leqslant \frac{\delta_i^w}{\delta_i^w + \varepsilon}.\end{aligned}$$

Hence,

$$\begin{aligned}|\lambda_p| &\leqslant -1 + 2\frac{\delta_i^w}{\delta_i^w + \varepsilon} = 1 - 2\frac{\varepsilon}{\delta_i^w + \varepsilon} \\ &\leqslant 1 - 2\frac{\varepsilon}{\Delta^w + \varepsilon}. \qquad \square\end{aligned}$$

Having these upper bounds for the second largest eigenvalue and for the absolute value of the smallest eigenvalue, the following result that fixes an upper bound for the convergence factor of a GDM holds.

Corollary 12. *Let $H = (G, l, c, w)$ a heterogeneous model, $M(\varepsilon)$ a GDM that uses the scalars defined in Eq* (3.1) *and*

$$\varepsilon_0 = 2e(G) w_{\min} \frac{c_{\min}}{c_{\max}} \sin^2\left(\frac{\pi}{2p}\right).$$

Then,

$$\lambda_{M(\varepsilon)} \leqslant 1 - 4\, e(G) \frac{w_{\min}}{\Delta^w + \varepsilon} \frac{c_{\min}}{c_{\max}} \sin^2\left(\frac{\pi}{2p}\right) \quad \forall \varepsilon \geqslant \varepsilon_0. \tag{3.5}$$

Proof. The result follows directly from Eqs. (3.2) and (3.3). □

The upper bound fixed by the above inequality is minimized when $\varepsilon = \varepsilon_0$. Therefore, the diffusion based on $M(\varepsilon_0)$ may converge faster than an algorithm based on $M(\varepsilon)$, with $\varepsilon > \varepsilon_0$.

Theorem 13. *The number of iterations needed by the GDA based on* $M(\varepsilon_0)$ *to balance the system belongs to* $O((c_{\max}\Delta^w p^2)/(c_{\min}\, w_{\min}\, e(G))$.

Proof. Suppose that we want to reduce the load imbalance to τ, sufficiently small so that the workload distribution, after a number of steps n, is close to the fair distribution. On the basis of Eq. (3.10), it is sufficient to take n so that

$$||l^{(n)} - \bar{l}||_{2,D^{-1/2}} \leqslant \left(1 - 4e(G) \frac{w_{\min}}{\Delta^w + \varepsilon_0} \frac{c_{\min}}{c_{\max}} \sin^2\left(\frac{\pi}{2p}\right)\right)^n ||l^{(0)} - \bar{l}||_{2,D^{-1/2}} \simeq \tau.$$

As for large p one has

$$\ln(1 - 2\varepsilon_0/(\Delta^w + \varepsilon_0)) \simeq 2\varepsilon_0/(\Delta^w + \varepsilon_0) \quad \text{and} \quad \sin\left(\frac{\pi}{2p}\right) \simeq \frac{\pi}{2p},$$

we conclude that the maximum number of steps necessary to reduce the load imbalance to τ is

$$\begin{aligned} n &\simeq \ln\left(\frac{1}{\tau}||l^{(0)} - \bar{l}||_{2,D^{-1/2}}\right) \frac{1}{\pi^2 e(G)} \frac{c_{\max}}{c_{\min}} \frac{\Delta^w + \varepsilon_0}{w_{\min}} p^2 \\ &\simeq \ln\left(\frac{1}{\tau}||l^{(0)} - \bar{l}||_{2,D^{-1/2}}\right) \frac{1}{\pi^2 e(G)} \frac{c_{\max}}{c_{\min}} \frac{\Delta^w}{w_{\min}} p^2 \\ &\quad + \frac{1}{2} \ln\left(\frac{1}{\tau}||l^{(0)} - \bar{l}||_{2,D^{-1/2}}\right). \qquad \square \end{aligned}$$

The above estimation is consistent with the result obtained in the homogeneous case by Boillat, who showed that the diffusion needs $O(p^2)$ communication steps [1]. In the homogeneous case, when $c_i = c_j$ for all $i \neq j$ and $w_k = 1$, for all $k = 1, q$, the diffusion matrix proposed in [1] can be expressed in our terms as $M(1)$. Particularizing the results obtained above, a homogeneous diffusion based on $M(\varepsilon_0)$ may converge faster than the algorithm given in the cited paper, as ε_0 improves the upper bound for the convergence factor given in Corollary 12. This is also confirmed partly by the theoretical results obtained in [30] and by the tests that were done.

3.1. Comparison with other approaches

A hydrodynamic approach was proposed by Hui and Chanson [17,18] to solve the load balancing problem in a heterogeneous system. However, they assumed only different processing performances, the communication being considered uniform. The computing nodes were viewed as cylinders of different capacities, relied by pipes with the same characteristics. Naturally, the liquid flow through a pipe can be assimilated to a load transfer between processors. The volume of liquid contained by a cylinder was assimilated to the load of a processor and the cross-sectional area of a cylinder to the speed of a processor. The authors used the concept of *global potential energy* to analyze their algorithm. Denoting the liquid volume of a node n_i by l_i and its cross-sectional area by c_i, the *height* of a node n_i was defined as $h_i = l_i/c_i$, *the potential energy of the liquid column* in n_i as $\mathrm{PE}(n_i) = 1/2 c_i h_i^2$, and the global potential energy as $\mathrm{GPE}(G) = \sum_{n_i \in V} \mathrm{PE}(n_i)$. The authors showed that the fair state, which is characterized by equal heights of the liquid columns, is achieved when $\mathrm{GPE}(G)$ is minimal. If $\mathrm{GPE}^{(n)}$ denotes the global potential energy of the system after the nth step and $\mathrm{GPE}^{\min}$ the minimum global potential energy corresponding to the equilibrium state, the following relation takes place:

$$\mathrm{GPE}^{(n)} < \eta^n \mathrm{GPE}^{(0)} + (1 - \eta^n)\mathrm{GPE}^{\min} \quad \text{for all } n \geqslant 0, \tag{3.6}$$

where

$$\eta = 1 - \frac{\gamma^2}{2(p-1)\mathrm{diam}(G)^2} \frac{c_{\min}^2}{c_{\max} \sum_i c_i}$$

and $\gamma \in (0, 1]$. The variable η indicates the convergence rate. The parameter γ was referred to as the *balancing factor*.

Our key observation is that the difference $\mathrm{GPE}^{(n)} - \mathrm{GPE}^{\min}$ is nothing but the square of the distance induced by the weighted 2-norm, between the workload vector corresponding to the current state and the workload vector corresponding to the fair state. Indeed, the quantities $\mathrm{GPE}^{(n)}$ and $\mathrm{GPE}^{\min}$ can be rewritten as

$$\mathrm{GPE}^{(n)} = \frac{1}{2} \sum_i \frac{(l_i^{(n)})^2}{c_i}$$

and

$$\mathrm{GPE}^{\min} = \frac{1}{2} \sum_i \frac{\bar{l}_i^2}{c_i}.$$

On the other hand, we have

$$||l^{(n)} - \bar{l}||^2_{2,D^{-1/2}} = \sum_i \frac{(l_i^{(n)} - \bar{l}_i)^2}{c_i} = \sum_i \frac{(l_i^{(n)})^2}{c_i} + \sum_i \frac{\bar{l}_i^2}{c_i} - 2\sum_i \frac{l_i^{(n)}\bar{l}_i}{c_i}$$

and

$$\sum_i \frac{l_i^{(n)}\bar{l}_i}{c_i} = \sum_i l_i^{(n)} \sum_j l_j^{(0)} = \left(\sum_j l_j^{(0)}\right)^2 = \sum_i c_i\left(\sum_j l_j^{(0)}\right)^2 = \sum_i \frac{c_i^2(\sum_j l_j^{(0)})^2}{c_i} = \sum_i \frac{\bar{l}_i^2}{c_i}.$$

It follows that

$$||l^{(n)} - \bar{l}||^2_{2,D^{-1/2}} = \sum_i \frac{(l_i^{(n)})^2}{c_i} - \sum_i \frac{\bar{l}_i^2}{c_i} = 2(\mathrm{GPE}^{(n)} - \mathrm{GPE}^{\min}).$$

Therefore, inequality (3.6) can be rewritten as

$$||l^{(n)} - \bar{l}||^2_{2,D^{-1/2}} < \eta^n ||l^{(0)} - \bar{l}||^2_{2,D^{-1/2}}. \tag{3.7}$$

On the other hand, as it was shown in Section 2.2, the generalized diffusion algorithm satisfies the relation

$$||l^{(n)} - \bar{l}||^2_{2,D^{-1/2}} \leqslant \lambda_M^{2n} ||l^{(0)} - \bar{l}||^2_{2,D^{-1/2}}. \tag{3.8}$$

We show that a generalized diffusion algorithm has a better convergence factor.

Theorem 14. *In a heterogeneous environment $H = (G, l, c, w)$ with $\mathrm{diam}(G) \geqslant 2$ and $w_k = 1$ for all $k = 1, q$, a generalized diffusion algorithm based on $1/2(I + M(\varepsilon_0))$ has a smaller convergence factor than the hydrodynamic algorithm.*

Proof. Because $w_k = 1$ for all edges, we have that $\delta_i^w = \delta_i$ for all i and $\Delta^w = \Delta$. The reason for considering $M' = 1/2(I + M(\varepsilon_0))$ is that this matrix has only nonnegative eigenvalues and in this case $\lambda_{M'} = \lambda_2(M') = 1/2(1 + \lambda_2(M(\varepsilon_0))$. We shall show that the convergence factor of a diffusion based on $1/2(I + M(\varepsilon_0))$ is always smaller than η. For this purpose we use Theorem 5, which states that

$$\lambda_2(M(\varepsilon_0)) \leqslant 1 - \frac{\theta(M(\varepsilon_0))}{\mathrm{diam}(G)}. \tag{3.9}$$

On the other hand, $m(\varepsilon_0)_{ij}c_j = \min\{c_i/(\delta_i + \varepsilon_0), c_j/(\delta_j + \varepsilon_0)\}$, for all edges $\{i,j\}$ and $\theta(M(\varepsilon_0)) \geqslant c_{\min}/(\Delta + \varepsilon_0)$. It follows that

$$\lambda_2(M(\varepsilon_0)) \leqslant 1 - \frac{c_{\min}}{\mathrm{diam}(G)\,(\Delta + \varepsilon_0)}. \tag{3.10}$$

One should prove that $\lambda_{M'}^2 \leqslant \eta$, i.e. $1/4\,(1 + \lambda_2(M(\varepsilon_0))^2 \leqslant \eta$. It is sufficient to show that

$$1 - \frac{1}{2}\frac{c_{\min}}{\mathrm{diam}(G)\,(\Delta + \varepsilon_0)} \leqslant 1 - \frac{\gamma^2}{2(p-1)\mathrm{diam}(G)^2}\frac{c_{\min}^2}{c_{\max}\sum_i c_i}.$$

The last inequality is obviously true since $\sum_i c_i = 1, c_{\max} \geqslant c_{\min}, \gamma \in (0, 1]$ and $(p-1)\mathrm{diam}(G) \geqslant \Delta + \varepsilon_0$.

In fact, we showed something more, that $\lambda_{M'} < \eta$; as $\lambda_{M'}$ is subunitary, obviously, $\lambda_{M'}^2 < \lambda_{M'}$. Looking at (3.7) and (3.8) we conclude that the considered GDA potentially converges faster than the hydrodynamic algorithm, as it has a smaller convergence factor. □

4. Experimental results

The complexity bounds formulated in the previous section are tight in the sense that for some common topologies $M(\varepsilon_0)$ is very close to the optimum. For example, for a ring, Xu and Lau [30] found in the homogeneous case optimal diffusion matrices that can be expressed in our terms as $M(\varepsilon_{\mathrm{opt}})$, with

$$\varepsilon_{\mathrm{opt}} = \begin{cases} 2\sin^2\left(\dfrac{\pi}{p}\right) & \text{if } p \text{ is even,} \\ 2\sin\left(\dfrac{\pi}{2p}\right)\sin\left(\dfrac{3\pi}{2p}\right) & \text{if } p \text{ is odd.} \end{cases}$$

In this case, $\varepsilon_0 = 2\sin^2(\pi/2p)$, and one can notice that this is very close to $\varepsilon_{\mathrm{opt}}$ (they are both in $\Theta(1/p^2)$).

For a path graph with p vertices, in the homogeneous case,

$$M(\varepsilon_0) = I - \frac{1}{2 + \varepsilon_0}L$$

with L the corresponding Laplacian. Therefore, $\lambda_{M(\varepsilon_0)} = 1 - 4/(2 + \varepsilon_0)\sin^2(\pi/2p)$. On the other hand, inequality (2.9) reads in this case as

$$\lambda_{M(\varepsilon_0)} \leqslant 1 - 4\frac{1}{2 + \varepsilon_0}\sin^2\left(\frac{\pi}{2p}\right).$$

Therefore, the upper bound fixed in Eq. (2.9) is reached, i.e. it gives the exact value of the convergence factor of the diffusion on a path. For p-dimensional meshes, optimal matrices depending on a single parameter were indicated by Xu and Lau [30]. Expressed in our terms, these matrices coincide with $M(0)$ in all the cases. However, when the speeds and the communication parameters differ, it is difficult to give analytical characterizations for the convergence factor, even for simple cases. From this point of view, the general bounds obtained in Section 2.3 are useful.

We simulated and tested several cases of heterogeneous diffusion. We considered different artificially generated topologies, computing speeds, communication weights and initial loads. We used generalized diffusion matrices of type

$$M = I - AS(\varepsilon)WA^TD^{-1}$$

with $S(\varepsilon) = \mathrm{diag}(s_1(\varepsilon), \ldots, s_q(\varepsilon))$. More concretely, the diffusion matrices $M^i = I - AS^i(\varepsilon)WA^TD^{-1}$, $i = 1,5$, were considered, with the vectors of scalars s^i defined as follows:

$$s_k^1(\varepsilon) = \min\left\{\frac{c_i}{\delta_i^w + \varepsilon}, \frac{c_j}{\delta_j^w + \varepsilon}\right\}, \quad \forall k : e_k = \{i,j\} \quad \text{and} \quad e_k \in E, \tag{4.1}$$

$$s_k^2(\varepsilon) = \frac{c_{\min}}{\Delta^w + \varepsilon} \quad \forall k : e_k = \{i,j\} \quad \text{and} \quad e_k \in E, \tag{4.2}$$

$$s_k^3(\varepsilon) = \sqrt{\frac{c_{\min}}{c_{\max}}}\frac{\sqrt{c_i\, c_j}}{\Delta^w + \varepsilon} \quad \forall k : e_k = \{i,j\} \quad \text{and} \quad e_k \in E, \tag{4.3}$$

$$s_k^4(\varepsilon) = \frac{c_ic_j}{c_{\max}}\frac{1}{\Delta^w + \varepsilon} \quad \forall k : e_k = \{i,j\} \quad \text{and} \quad e_k \in E, \tag{4.4}$$

$$s_k^5(\varepsilon) = \frac{c_ic_j}{w_k} \quad \forall k : e_k = \{i,j\} \quad \text{and} \quad e_k \in E. \tag{4.5}$$

The first is a generalization of the case investigated by Boillat [1]. The diffusion matrices M^2, M^3 and M^4 are generalizations of the homogeneous case when all the entries of the diffusion matrix corresponding to edges of the communication graph are set to the same value $\alpha = 1/(\Delta + 1)$. The tests performed with these algorithms showed that the diffusion that uses $M^1(\varepsilon_0)$ takes generally the fewest number of iterations. Of particular interest is the case when the communication topology is a path, because it is well known that on graphs of this type the diffusion has the slowest convergence. Tables 1 and 2 report results for the cases when the ratio between the maximum and the minimum speed, $\alpha = c_{\max}/c_{\min}$, is in the set $\{3, 5, 10\}, w_k = 1$, for all edges $e_k, \varepsilon \in \{1, \varepsilon_0\}$, when the number of processors is 16, 32 or 64 and for a fixed random load distribution. The number of iterations performed until $||l^{(n)} - \bar{l}||_2 < 0.1$ of the diffusion algorithms defined by Eqs. (4.1)–(4.5) are given in these tables. Clearly, the convergence of any GDA deteriorates when the number of processors increase, when the ratio between the maximum speed and the minimum speed is large, when the communication graph has a small edge connectivity or a large diameter. The worst diffusion showed to be that defined by Eq. (4.5); it shows how important is the choice of a diffusion matrix in a heterogeneous context and it serves as a very bad example.

In a second step, we performed tests with the diffusion algorithms that use $M^1(1), M^1(\varepsilon_0)$ and $M^6 = I - AS^6WA^TD^{-1}$, where

$$s_k^6 = \frac{2}{\mu_2(\mathscr{L}) + \mu_p(\mathscr{L})} \quad \forall k : e_k \in E. \tag{4.6}$$

Table 1
Number of steps until convergence of the GDAs that are using the diffusion matrices $M^1, \ldots, M^5$, on two-dimensional meshes with $4 \times 4, 4 \times 8$ and 8×8 vertices

p	α	ε	Number of iterations				
			M^1	M^2	M^3	M^4	M^5
16	3	1.0000	115	167	164	170	357
		0.0384	86	134	131	136	357
	5	1.0000	172	285	300	358	662
		0.0384	129	229	241	288	662
	10	1.0000	127	348	298	216	331
		0.0384	97	280	239	173	331
32	3	1.0000	473	668	640	621	2623
		0.0096	364	535	512	497	2623
	5	1.0000	419	960	790	497	2623
		0.0096	323	769	632	500	2337
	10	1.0000	478	1822	1298	1008	3398
		0.0096	369	1460	1040	807	3398
64	3	1.0000	466	651	608	552	4705
		0.0024	365	520	486	441	4705
	5	1.0000	529	978	814	662	5061
		0.0024	415	782	651	529	5061
	10	1.0000	669	1736	1290	1271	8670
		0.0024	524	1388	1032	1017	8670

Table 2
Number of steps until convergence of the GDAs that are using the diffusion matrices $M^1, \ldots, M^5$, on paths with 16, 32 and 64 vertices

p	α	ε	Number of iterations				
			M^1	M^2	M^3	M^4	M^5
16	3	1.0000	1262	1507	1479	1519	5245
		0.0192	848	1013	994	1021	5245
	5	1.0000	1399	2241	2094	2288	7025
		0.0192	940	1507	1408	1538	7025
	10	1.0000	1330	3021	2700	3733	9463
		0.0192	894	2032	1815	2511	9463
32	3	1.0000	4989	5985	5853	5994	41 981
		0.0048	3332	3998	3910	4004	41 981
	5	1.0000	5348	8751	8090	8695	53 930
		0.0048	3572	5846	5405	5809	53 930
	10	1.0000	6215	15 963	13 554	16 596	92 957
		0.0048	4152	10 666	9056	11 089	92 957
64	3	1.0000	20 161	24 211	23 654	24 202	341 570
		0.0012	13 447	16 149	15 777	16 143	341 570
	5	1.0000	22 340	36 368	33 686	36 245	459 160
		0.0012	14 901	24 258	22 470	24 177	459 160
	10	1.0000	24 453	63 855	53 385	64 234	728 035
		0.0012	16 310	42 594	35 610	42 847	728 035

The diffusion matrix M^6 defined as above is identical to that used by the generalized diffusion algorithm considered by Elsässer et al. [8]. This appears as a generalization of the algorithm proposed by Cybenko [4], when all the scalar parameters are set to the same value α, i.e. $s_k^6 = \alpha$, for all edges e_k. The best way to choose this value is as expressed in Eq. (4.6), as a function of the largest and the second smallest eigenvalue of the generalized Laplacian matrix. However, in a dynamic context the communication topology may change and the performance of the processors may vary between two successive calls of a dynamic load balancing algorithm. This means that also the diffusion matrix may change and a new application of the diffusion algorithm that uses M^6 requires in this case knowledge at runtime of these eigenvalues. Their computation introduces a supplementary overhead. The algorithm based on $M^1(\varepsilon_0)$ can be applied straightforward in these cases, requiring only minor update operations.

We proceeded to a comparison of the generalized diffusion algorithms that use the diffusion matrices $M^1(1)$, $M^1(\varepsilon_0)$ and M^6. We shall refer to these algorithms as GDA1, GDA0 and GDA6, respectively. These algorithms were compared in terms of number of iterations until convergence and in terms of convergence factor of the corresponding generalized diffusion matrices. The considered communication topologies were paths, two-dimensional meshes and two-dimensional tori, with a fixed dimension of size 4 and the other variable, and partial binary trees. We further considered two kinds of load distributions:

- *ONE*: all processors have the same load, excepting one whose load is $500 \times p$ times greater than of the others;
- *EQ*: the processors have affected the same load, equal to $200 \times p$.

Regarding the heterogeneity of the computational environment, the following test cases were considered:

- *HOMO*: the system is homogeneous both w.r.t. the processors' performances and the communication speeds.
- *HCUW*: heterogeneous environment characterized by different computing speeds and uniform communication. More specifically,
  - the speed of a processor was considered proportional to $r \bmod 4 + 1, r$ denoting the rank of the processor;
  - the weights on the edges were set to 1.
- *HCHW*: heterogeneous environment characterized by different computing speeds and non-uniform communication. More specifically,
  - the speed of a processor was considered proportional to $r \bmod 4 + 1, r$ denoting the rank of the processor;
  - for any edge $e_k = \{i,j\}$, the weight associated to it was set to $(i + j) \bmod 3 + 1$.

The used convergence criterion was $||l^{(n)} - \bar{l}||_2 < 0.1$. The scalar parameter used by the algorithm GDA6 (relying on M^6) was computed as a function of the largest and the second smallest eigenvalue of the generalized Laplacian, using the tool Mathematica 4.2.

With these settings, the number of iterations of the three algorithms for the considered communication topologies, computational environments and load distributions are as shown in Figs. 1,3–6, for paths, Figs. 2, 7–10, for two-dimensional meshes, Figs. 11–14, for two-dimensional tori and Figs. 15–18, for partial binary trees.

In the case of a communication topology of type path, in the homogeneous case, the tests showed that for a load distribution of type ONE, the algorithms GDA0 and GDA6 perform identically, as Fig. 1 illustrates. This must be indeed the case, as the scalar parameters are identical in $M(\varepsilon_0)$ and this is the optimal matrix depending on a single parameter in the case of a toplogy of type path, identical therefore to M^6. However, in the case of two-dimensional meshes (with a fixed dimension of size 4), GDA0 uses nonidentical scalar parameters and it was observed that it performs slightly better than GDA6, as shown in Fig. 2. Naturally, for a load distribution of type EQ, all algorithms performed 0

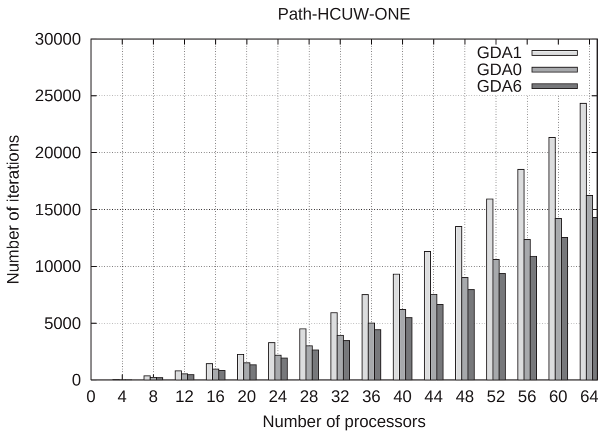

Fig. 3. Number of iterations of GDA1, GDA0 and GDA6 on paths with up to 64 vertices in the case HCUW-ONE.

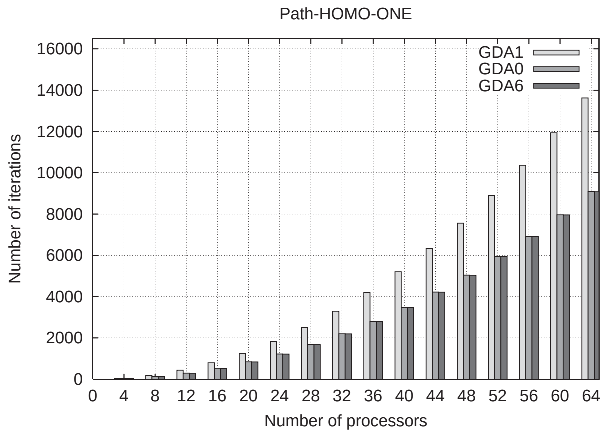

Fig. 1. Number of iterations of GDA1, GDA0 and GDA6 on paths with up to 64 vertices in the case HOMO-ONE.

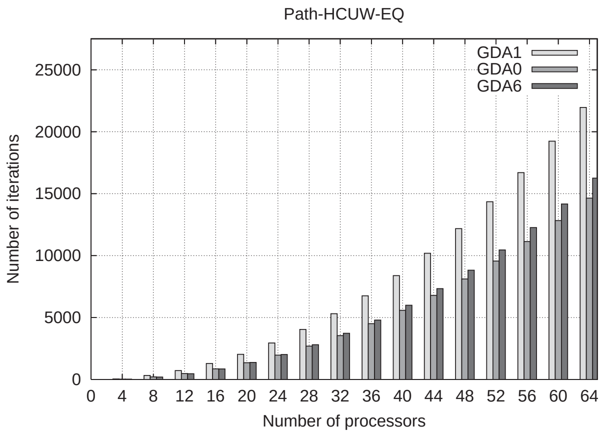

Fig. 4. Number of iterations of GDA1, GDA0 and GDA6 on paths with up to 64 vertices in the case HCUW-EQ.

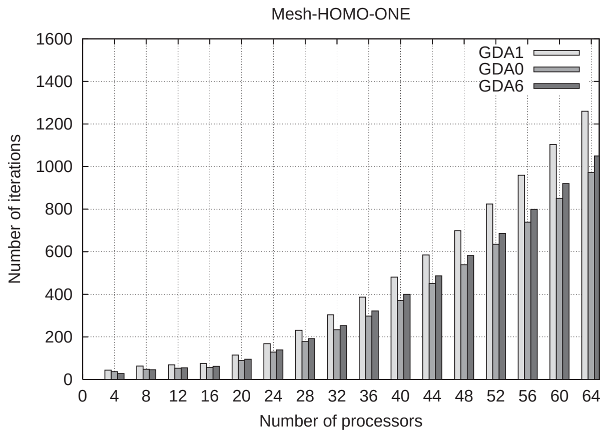

Fig. 2. Number of iterations of GDA1, GDA0 and GDA6 on meshes with up to 64 vertices in the case HOMO-ONE.

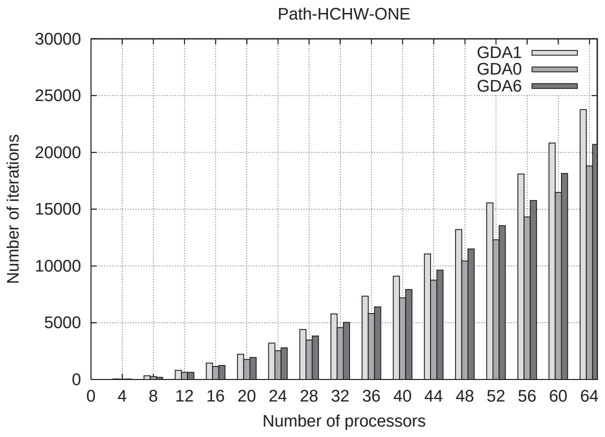

Fig. 5. Number of iterations of GDA1, GDA0 and GDA6 on paths with up to 64 vertices in the case HCHW-ONE.

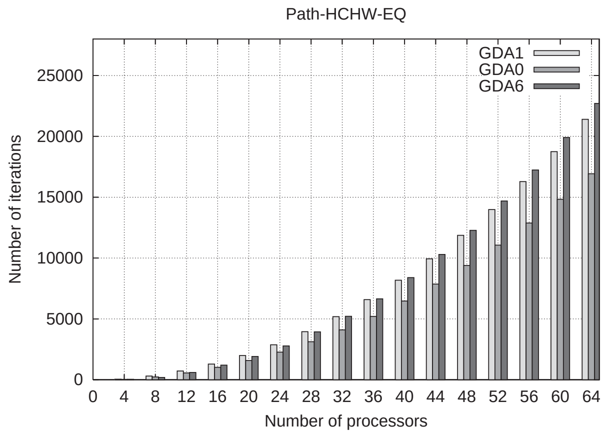

Fig. 6. Number of iterations of GDA1, GDA0 and GDA6 on paths with up to 64 vertices in the case HCHW-EQ.

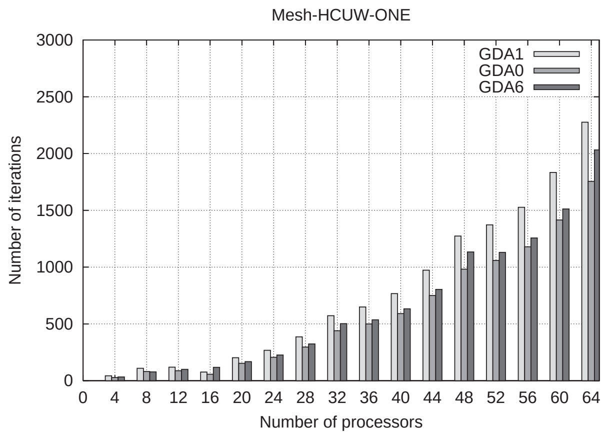

Fig. 7. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional meshes with up to 64 vertices in the case HCUW-ONE.

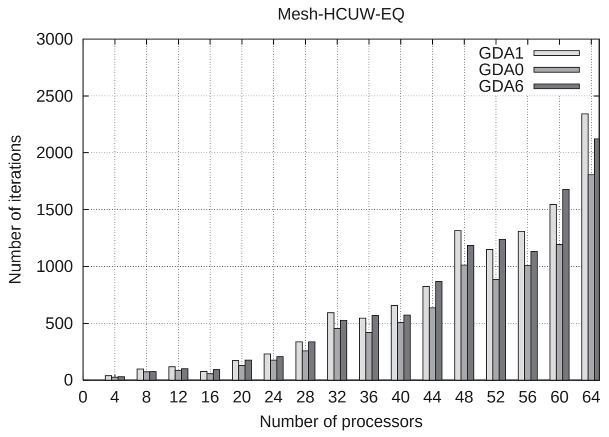

Fig. 8. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional meshes with up to 64 vertices in the case HCUW-EQ.

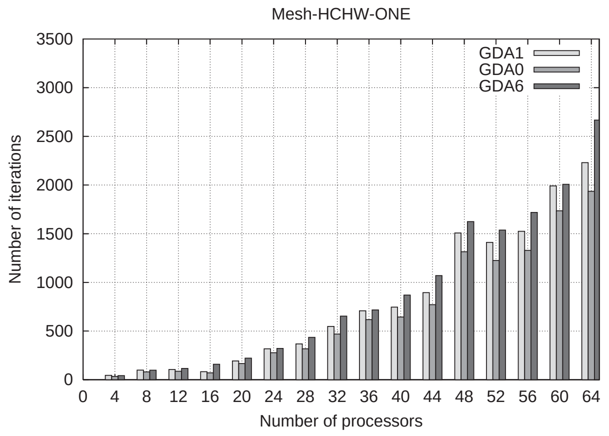

Fig. 9. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional meshes with up to 64 vertices in the case HCHW-ONE.

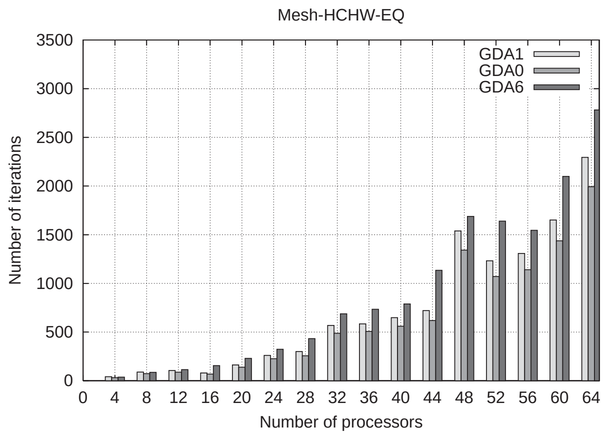

Fig. 10. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional meshes with up to 64 vertices in the case HCHW-EQ.

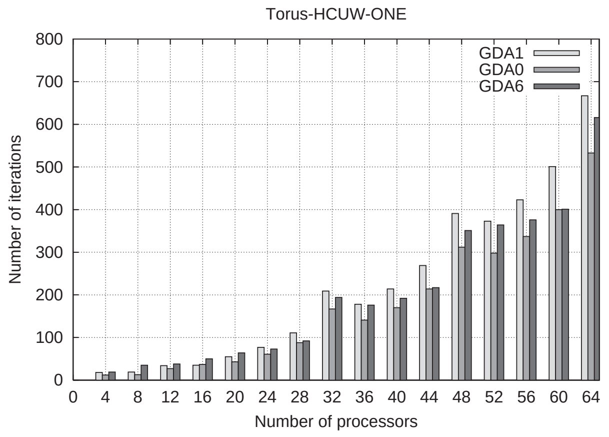

Fig. 11. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional tori with up to 64 vertices in the case HCUW-ONE.

iterations in a homogeneous environment, as the loads were already balanced.

Also in the case of a path, but in a heterogeneous computational environment of type HCUW, for a load distribution of type ONE it was observed that GDA6 performs better than GDA0 and GDA1, as Fig. 3 illustrates. In all the other cases, for a topology of type path, it was observed that GDA0 performs better than GDA6 and GDA1, as Figs. 4–6 show. In the HCHW case, for a load distribution of type EQ it was observed

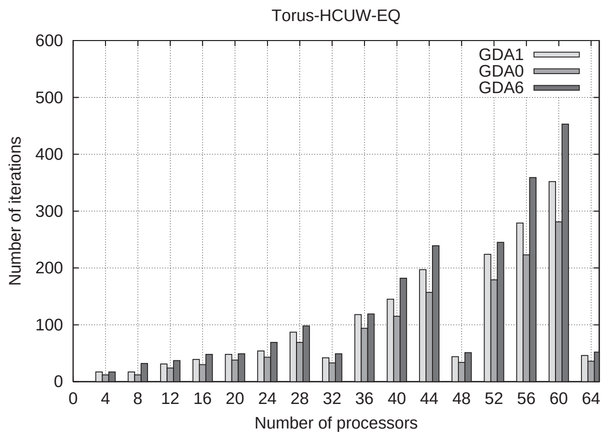

Fig. 12. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional tori with up to 64 vertices in the case HCUW-EQ.

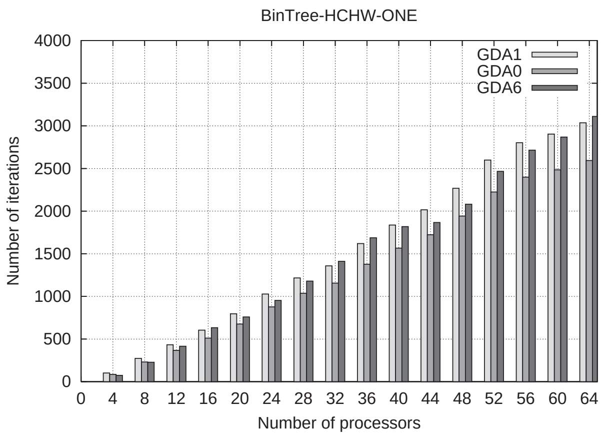

Fig. 15. Number of iterations of GDA1, GDA0 and GDA6 on partial binary trees with up to 64 vertices in the case HCHW-ONE.

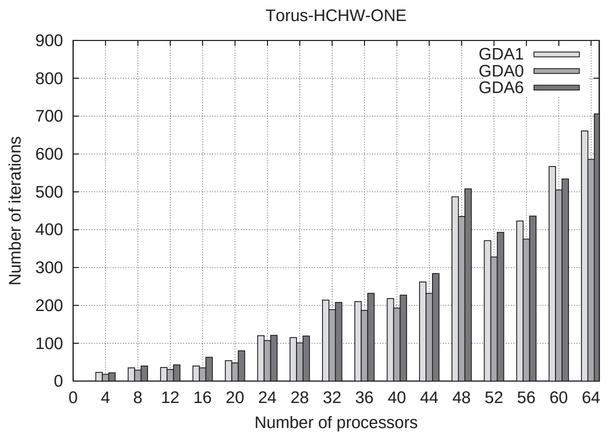

Fig. 13. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional tori with up to 64 vertices in the case HCHW-ONE.

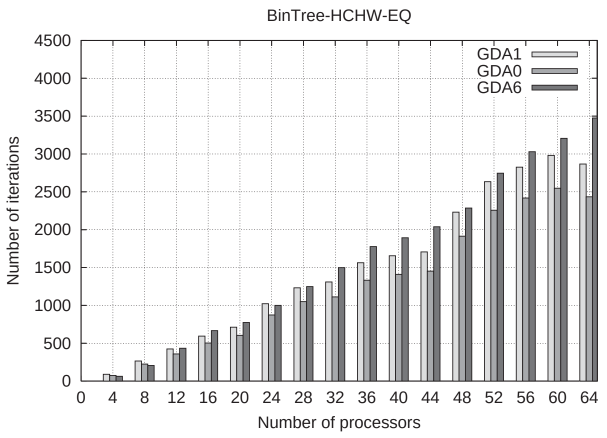

Fig. 16. Number of iterations of GDA1, GDA0 and GDA6 on partial binary trees with up to 64 vertices in the case HCHW-EQ.

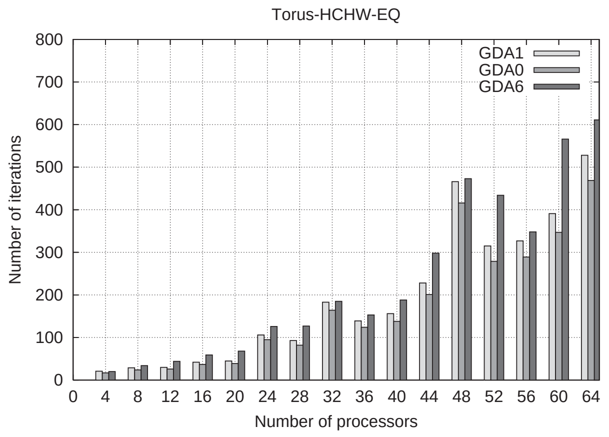

Fig. 14. Number of iterations of GDA1, GDA0 and GDA6 on two-dimensional tori with up to 64 vertices in the case HCHW-EQ.

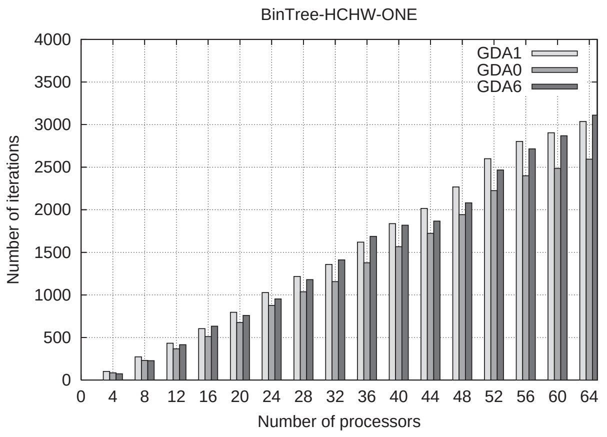

Fig. 17. Number of iterations of GDA1, GDA0 and GDA6 on partial binary trees with up to 64 vertices in the case HCHW-ONE.

that GDA6 performs even worse than GDA1, as Fig. 6 shows. It should be noted that both in the homogeneous case and the heterogeneous case the tests showed GDA0 to perform much better than GDA1, which means that ε_0 is a better choice than $\varepsilon = 1$, which was proposed by Boillat.

For a communication topology of type two-dimensional mesh, two-dimensional torus or partial binary

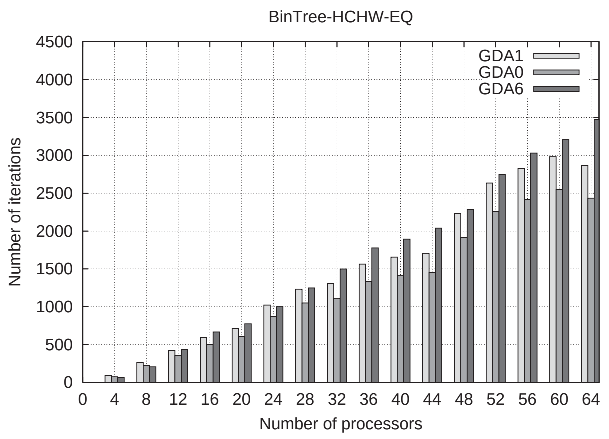

Fig. 18. Number of iterations of GDA1, GDA0 and GDA6 on partial binary trees with up to 64 vertices in the case HCHW-EQ.

tree, it was observed that in all the cases GDA0 performs better than GDA1 and GDA6 in the considered heterogenous contexts and for the considered load distributions, as illustrated in Figs. 7–18.

It was also observed that the considered diffusion algorithms perform faster on graphs that are characterized by a larger edge connectivity, a larger maximum degree and a smaller diameter than in the case of a path. This is in accordance with Theorem 5, which establishes an upper bound for the second largest eigenvalue of an arbitrary generalized diffusion matrix and which indicates that a GDA possibly converges faster on graphs with a larger edge connectivity and a smaller diameter. As the paths have the largest diameter among all the graphs with the same number of vertices, a GDA takes the largest number of steps to converge on this type of graphs compared to other topologies.

The tests also showed that for the considered topologies, computational environments and load distributions, it may be more appropriate to use multiple scalar parameters than a single scalar parameter, at least in the heterogeneous case in which both the communication and the processors have different performances. The restriction imposed by the algorithm GDA6, that all scalars are identical, may be the cause of the observed differences in terms of number of iterations and convergence factors. Of course, the accuracy with which the eigenvalues are computed, in our case using the tool Mathematica 4.2, may play also a role, especially when the diffusion matrices have large dimensions.

We further compared the three diffusion algorithms GDA1, GDA0 and GDA6 in terms of convergence factors of the corresponding generalized diffusion matrices.

In Table 3, the convergence factors of the algorithms GDA1, GDA0 and GDA6 are shown, in the case when the communication topologies are of type path and two-dimensional mesh, with up to 64 vertices, in a heterogeneous environment characterized by different processing speeds and uniform communication of type HCUW. These values were computed with Mathematica 4.2. In the case of a path topology, the convergence factor of the algorithm GDA6 is slightly smaller than that of the algorithm GDA0. As Fig. 3 illustrates, in this case the algorithm GDA6 takes a fewer number of iterations than GDA0 for a load distribution of type ONE and a greater number of iterations for a load distribution of type EQ. It should be remarked that the differences in terms of number of iterations between GDA6 and GDA0 are much smaller than between GDA6 and GDA1. In the case of a communication topology of type two-dimensional mesh, as can be seen in Table 3, GDA0 has a smaller convergence than GDA6. The tests showed, as expected, that the algorithm GDA0 takes fewer iterations to converge than GDA6 as can be seen in Figs. 7 and 8.

In Table 4, the convergence factors of the algorithms GDA1, GDA0 and GDA6 are shown, for paths and two-dimensional meshes with up to 64 vertices, in a heterogeneous environment characterized by different processing speeds and nonuniform communication of type HCHW. In this case, it was remarked that GDA0 always has a smaller convergence factor than GDA6 and that the last one has often greater values than the convergence factor of GDA1. The results shown in Table 4 are in accordance with the results illustrated in Figs. 5, 6, 9 and 10. In a heterogeneous environment of type HCHW and for the considered load distributions, GDA0 always performed a smaller number of iterations than GDA6 and GDA1.

5. Conclusions

The diffusion algorithms have been an active research topic in the area of parallel computing that aroused the interest of many researchers. In the last years, a number of important related contributions have been done. This paper aims to add some complementary results to the already existing ones. We dealt with aspects of the generalization of the diffusion algorithms for the case when the computational environment is heterogeneous both with respect to the processors' performances and the communication speeds. An explicit formula for the flow generated by the generalized diffusion algorithms was given. It was shown that this flow has an important property, that it is the projection of all other balancing flows in the same heterogeneous environment. The result of minimality with respect to a weighted 2-norm of the generated balancing flow, already known in homogeneous environments, appears thus as natural consequence of this result.

A generalization of the variant of diffusion proposed by Boillat [1] was investigated. Estimations were given for the maximum number of steps that such an iterative

Table 3
Convergence factors of GDA1, GDA0 and GDA6 on paths and two-dimensional meshes with up to 64 vertices, in the HCUW case

p	Path			Mesh		
	GDA1	GDA0	GDA6	GDA1	GDA0	GDA6
4	0.793967	0.707961	0.735786	0.793967	0.707961	0.735786
8	0.972906	0.959705	0.953002	0.913666	0.884954	0.880487
12	0.987453	0.981254	0.978519	0.921017	0.894285	0.904441
16	0.992853	0.989303	0.987807	0.868777	0.830583	0.916861
20	0.995400	0.993110	0.992162	0.951179	0.936059	0.941716
24	0.996796	0.995199	0.994545	0.962191	0.950989	0.954875
28	0.997642	0.996466	0.995986	0.973417	0.965448	0.967996
32	0.998192	0.997291	0.996924	0.981732	0.976327	0.979015
36	0.998570	0.997857	0.997568	0.983704	0.978873	0.980301
40	0.998841	0.998263	0.998030	0.986093	0.981998	0.983154
44	0.999042	0.998564	0.998371	0.988996	0.985741	0.986655
48	0.999195	0.998793	0.998631	0.991535	0.989030	0.990458
52	0.999314	0.998971	0.998833	0.992094	0.989761	0.990403
56	0.999408	0.999112	0.998996	0.992864	0.990765	0.991326
60	0.999484	0.999227	0.999123	0.994043	0.992286	0.992763
64	0.999546	0.999320	0.999230	0.995177	0.993748	0.994593

Table 4
Convergence factors of GDA1, GDA0 and GDA6 on paths and two-dimensional meshes with up to 64 vertices, in the HCHW case

p	Path			Mesh		
	GDA1	GDA0	GDA6	GDA1	GDA0	GDA6
4	0.804061	0.744105	0.781930	0.804061	0.744105	0.781930
8	0.970808	0.961725	0.949727	0.906022	0.885803	0.895722
12	0.987580	0.984210	0.984066	0.912215	0.895577	0.916015
16	0.992898	0.991072	0.991655	0.948963	0.940740	0.956303
20	0.995318	0.994107	0.994623	0.967840	0.963149	0.968041
24	0.996722	0.995860	0.996232	0.971980	0.967632	0.975959
28	0.997593	0.996960	0.997229	0.980934	0.977765	0.983879
32	0.998151	0.997666	0.997879	0.985044	0.982848	0.985215
36	0.998536	0.998151	0.998320	0.985044	0.982848	0.985215
40	0.998815	0.998503	0.998638	0.985709	0.983467	0.987679
44	0.999019	0.998761	0.998876	0.988016	0.986080	0.989972
48	0.999175	0.998958	0.999054	0.992842	0.991794	0.993327
52	0.999297	0.999112	0.999194	0.992342	0.991196	0.992934
56	0.999394	0.999235	0.999307	0.992862	0.991813	0.993659
60	0.999472	0.999333	0.999395	0.994511	0.993708	0.994545
64	0.999536	0.999414	0.999467	0.995080	0.994336	0.995872

process may take to balance the system. For this purpose, some general bounds were formulated for the second largest eigenvalue of a generalized diffusion matrix. These bounds were also used to show that there are generalized diffusion algorithms that theoretically converge faster than the hydrodynamic algorithm. The tests performed with several types of diffusion algorithms showed that the given algorithm performs generally better in the considered heterogeneous contexts. Finally, when compared to the algorithm GDA6 given in [8], which appears as a generalization of the variant of diffusion proposed in [4], it was shown that for common topologies like paths, two-dimensional meshes, two-dimensional tori or partial binary trees, the algorithm GDA0 has generally smaller convergence factors and takes fewer iterations to converge than GDA6 in the considered heterogeneous environments. This suggests that using diffusion matrices depending on different scalar parameters (which is the case of GDA0) may be more appropriate than using diffusion matrices depending on a single scalar parameter (which is the case of GDA6) in heterogeneous environments. The algorithm GDA0 can be applied straightforward in a dynamic context and does not use parameters that are dependent on the eigenvalues of the generalized Laplacian matrix.

Acknowledgments

We thank the anonymous referees for their constructive comments.

References

[1] J.E. Boillat, Load balancing and Poisson equation in a graph, Concurrency Practice Experience 2 (4) (1990) 289–313.

[2] F.R.K. Chung, Spectral Graph Theory, CBMS Lecture Notes, AMS Publication, American Mathematical Society, Providence, RI, 1996.

[3] F.R.K. Chung, P. Tetali, Isoperimetric inequalities for cartesian products of graphs, Probab. Combin. Comput. 7 (1998) 141–148.

[4] G. Cybenko, Dynamic load balancing for distributed memory multi-processors, J. Parallel Distrib. Comput. 7 (1989) 279–301.

[5] R. Diekmann, A. Frommer, B. Monien, Efficient schemes for nearest neighbor load balancing, Parallel Comput. 25 (7) (1999) 789–812.

[6] R. Diekmann, S. Muthukrishnan, M.V. Nayakkankuppam, Engineering diffusive load balancing algorithms using experiments, in: G. Bilardi, A. Ferreira, R. Lueling, J. Rolim (Eds.), Solving Irregulary Structured Problems in Parallel (IRREGULAR '97), Lecture Notes in Computer Science, Vol. 1253, Springer, Berlin, 1997, pp. 111–122.

[7] R. Elsässer, B. Monien, R. Preis, Diffusive load balancing schemes on heterogeneous networks, in: G. Bilardi, et al. (Eds.), 12th ACM Symposium on Parallel Algorithms and Architectures (SPAA), Vol. 1461, 2000, pp. 30–38.

[8] R. Elsässer, B. Monien, R. Preis, Diffusion schemes for load balancing on heterogeneous networks, Theory Comput. Systems 35 (2002) 305–320.

[9] M. Fiedler, Bounds for eigenvalues of doubly stochastic matrices, Linear Algebra Appl. 5 (1972) 299–310.

[10] M. Fiedler, A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory, Czechoslovak Math. J. 25 (100) (1975) 619–633.

[11] B. Hendrickson, K. Devine, Dynamic load balancing in computational mechanics, Comput. Methods Appl. Mech. Eng. 184 (2–4) (2000) 485–500.

[12] A. Heririch, S. Taylor, A parabolic load balancing method, in: Proceedings of the 24th International Conference on Parallel Processing, Vol. III, CRC Press, Urbana-Champaign, IL, August 1995, 192–202.

[13] G. Horton, A multi-level diffusion method for dynamic load balancing, Parallel Comput. 19 (1993) 209–218.

[14] Y.F. Hu, R.J. Blake, An improved diffusion algorithm for dynamic load balancing, Parallel Comput. 25 (1999) 417–444.

[15] Y.F. Hu, R.J. Blake, Load balancing for unstructured mesh applications, Parallel Distrib. Comput. Practice 2 (3) (1999).

[16] Y.F. Hu, R.J. Blake, D.R. Emerson, An optimal dynamic load balancing algorithm, Concurrency Practice Experience 10 (1998) 467–483.

[17] C.C. Hui, S.T. Chanson, Theoretical analysis of the heterogeneous dynamic load balancing problem using a hydro-dynamic approach, J. Parallel Distrib. Comput. 43 (1997) 139–146.

[18] C.C. Hui, S.T. Chanson, Hydrodynamic load balancing, IEEE Trans. Parallel Distrib. Systems 10 (11) (1999) 1118–1137.

[19] B. Litow, S.H. Hosseini, K. Vairavan, Performance characteristics of a load balancing algorithm, J. Parallel Distrib. Comput. 31 (2) (1995) 159–165.

[20] L. Lovasz, Random walks on graphs: a survey, Combinatorics Paul Erdos is Eighty 2 (1993) 1–46.

[21] C.D. Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, Philadelphia, PA, 2000.

[22] B. Mohar, Some applications of Laplace eigenvalues of graphs, Graph Symmetry Algebraic Methods Appl. 497 (1997) 225–275.

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

[24] T. Rotaru, H.-H. Nägeli, Heterogeneous dynamic load balancing, in: Grigoras et al. (Eds.), NATO Advanced Workshop on Advanced Environments, Tools and Applications for Cluster Computing, Lecture Notes in Computer Science, Vol. 2326, Mangalia, Springer, Berlin, September 2001, pp. 136–144.

[25] A. Sinclair, Improved bounds for mixing rates of markov chains and multicommodity flow, Combin. Probab. Comput. 1 (1992) 351–370.

[26] A. Sinclair, M.R. Jerrum, Approximate counting, uniform generation and rapidly mixing markov chains, Inform. Comput. 82 (1989) 93–133.

[27] R.S. Varga, Matrix Iterative Analysis, Series in Automatic Computation, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1962.

[28] J. Watts, S. Taylor, A practical approach to dynamic load balancing, IEEE Trans. Parallel Distrib. Systems 9 (1998) 235–248.

[29] M.H. Willebeeck-LeMair, A.P. Reeves, Strategies for dynamic load balancing on highly parallel computers, IEEE Trans. Parallel Distrib. Systems 4 (9) (1993) 1305–1336.

[30] C. Xu, F. Lau, Load Balancing in Parallel Computers Theory and Practice, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Dordrecht, 1997.