JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 24, 72–85 (1995)

The Generalized Dimension Exchange Method for Load Balancing in *k*-ary *n*-Cubes and Variants

CHENG-ZHONG XU*,[1] AND FRANCIS C. M. LAU†

*Department of Computer Science, Shantou University, People's Republic of China; and †Department of Computer Science, The University of Hong Kong, Hong Kong

The generalized dimension exchange (GDE) method is a fully distributed load balancing method that operates in a relaxation fashion for multicomputers with a direct communication network. It is parameterized by an exchange parameter λ that governs the splitting of load between a pair of directly connected processors during load balancing. An optimal λ would lead to the fastest convergence of the balancing process. Previous work has resulted in the optimal λ for the binary *n*-cubes. In this paper, we derive the optimal λ's for the *k*-ary *n*-cube network and its variants—the ring, the torus, the chain, and the mesh. We establish the relationships between the optimal convergence rates of the method when applied to these structures, and conclude that the GDE method favors high-dimensional *k*-ary *n*-cubes. We also reveal the superiority of the GDE method to another relaxation-based method, the diffusion method. We further show through statistical stimulations that the optimal λ's do speed up the GDE balancing procedure significantly. Because of its simplicity, the method is readily implementable. We report on the implementation of the method in two data-parallel computations in which the improvement in performance due to GDE balancing is substantial. © 1995 Academic Press, Inc.

1. INTRODUCTION

We consider the problem of dynamic load balancing in multicomputers. Multicomputers are a class of parallel machines that are composed of many autonomous processors interconnected by a communication network [1]. The processors do not share any memory and they conmmunicate among themselves via message passing. From time to time, the workload that is spread across the processors is found to be in an unbalanced state; load balancing is then initiated to balance the workload. Dimension exchange (DE) is one of the few distributed load balancing methods that operate in a relaxation fashion for point-to-point networks (a detailed survey can be found in [20]). With the DE method, an instance of load balancing is carried out as a sequence of "sweeps." During each sweep, a processor compares successively its workload with that of each of its nearest neighbors; following each such comparison, an exchange operation is executed to equalize the workload between this node and the neighboring node concerned. Alternatively, instead of exchanging workloads on-the-fly, the load balancing procedure can be divided into two phases: in the first phase, DE is employed to work out the revised load "indices" that correspond to a balanced state; then, in the second phase, the actual load migrations take place. This makes the method more applicable to situations in which the workload involves large amounts of data.

The DE method was initially intensively studied in hypercube-structured multicomputers [14, 15, 5]. Since the set of neighbors of a processor corresponds exactly to the dimensions of the hypercube, a sweep of the iterative process is equal to going through all the dimensions once. Cybenko proved that, regardless of the order in which these dimensions are considered in a sweep, this simple load balancing method yields a uniform distribution from any initial workload distribution after one sweep [5]. He also revealed the superiority of the DE method to another relaxation-based method, the diffusion method [3, 5], when applied to hypercubes. This theoretical result was supported in part by the experiment carried out by Willebeek-LeMair and Reeves [18].

The DE method is not limited to hypercube structures. Hosseini *et al.* analyzed the method as applied to arbitrary structures based on edge coloring of undirected graphs [8]. With edge coloring, the edges of a given graph are colored with some minimum number of colors such that no two adjoining edges are of the same color. A "dimension" is then defined to be the collection of all edges of the same color. Obviously, an *n*-dimensional hypercube can be colored with a minimum of *n* colors. During each iteration sweep, all dimensions (colors) are considered in turn. Since no two adjoining edges have the same color, each processor needs to deal with only one neighbor at a time during a sweep. Clearly, for an arbitrary structure, the DE method can no longer yield a uniform workload distribution in a single sweep. Nonetheless, Hosseini *et al.* showed that, given any arbitrary structure, the DE method converges eventually to a uniform distribution [8].

The DE method is characterized by "equal splitting" of workload between a pair of neighboring processors at

[1] Author's research supported in part by a Li ka Shing Scholarship.

0743-7315/95 $6.00

every comparison, which was shown to be optimal in hypercube structures but not necessarily so in other structures through our analysis [21]. In that paper, we generalized the DE method by adding an *exchange parameter* to govern the amount of workload (instead of always half) exchanged at every step. This method is called the *generalized dimension exchange* (GDE) method. We modeled this generalized DE method using a matrix iterative approach and derived the necessary and sufficient conditions for its convergence.

In this paper, we continue our analysis of the GDE method as applied to the family of k-ary n-cubes which include the ring, the chain, the torus, and the mesh. A k-ary n-cube is a structure with n dimensions, k nodes in each dimensions [6, 10]. The ring and the hypercube are special cases of the k-ary n-cube. A ring of k nodes is a k-ary 1-cube, and an n-dimensional hypercube is a 2-ary n-cube. The n-dimensional torus is a generalization of the k-ary n-cube, which allows different numbers of nodes in different dimensions. Taking a ring and a torus and strip them of all the end-round connections, we get a chain and a ring, respectively. We limit our scope to these structures because they are the most popular choices of topologies in commercial parallel computers [10, 13, 16]. Examples include the hypercube-structured Intel iPSC/860 and NCUBE/2, the mesh-structured Intel Paragon, Intel Touchstone Delta, iWarp, and Ametek 2010.

The main contribution of this paper is the derivation of the optimal exchange parameters in closed form for the family of k-ary n-cubes. The optimal solutions for these structures are of considerable value because there is this real need in practical situations of choosing an exchange parameter that would lead to the fastest convergence of the balancing procedure. A preview of these optimal parameters without proofs has been included in our previous paper [21]. A subset of the proofs, which are based on circulant matrix theory [7], will be presented in this paper. We capitalize on the modeling power of circulant matrices, which is most evident in cases in which the structures concerned can be recursively defined.

The other important contributions of this paper include the establishment of the relationships between the convergence rates of these structures and a proof of the superiority of the GDE method to the diffusion method. The latter is with respect to the convergence rates of the two methods when applied to the family of k-ary n-cubes. The matrix analysis reveals the asymptomatic convergence rates but sheds little light on the exact number of sweeps needed for balancing. Therefore, we use statistical simulations to obtain the actual number of sweeps required by GDE balancing in these structures. This number turns out to be encouragingly small in all the cases we tried when using the optimal parameters we derived. This adds a lot of weight to the practicality of the GDE method. In fact, the method has been employed in the implementations of two realistic data-parallel computations, and the improvement over the versions without load balancing is substantial. We give a brief report on these implementations.

The rest of this paper is organized as follows. In Section 2, we review the GDE method and its convergence properties for the general case. In Section 3, we analyze the GDE method for the k-ary n-cube and its variants, and derive their optimal exchange parameters. In Section 4, we make a comparison between the GDE method and the diffusion method. Section 5 reports on the results of a statistical simulation concerning the number of iteration sweeps as well as findings from practical implementations. We conclude in Section 6 with a summary of the results and discussion of further work.

2. THE GDE METHOD

The model of the underlying system and computation assumed in this study is similar to that in [3, 5, 8, 21]. Specifically, the multicomputer we consider consists of a finite set of homogeneous processors interconnected, as a point-to-point network. The communication links are bidirectional and the processors interact synchronously with one another. We represent such a system by a simple connected graph $G = (V, E)$, where V is a set of processors labeled 1 through N, and $E \subseteq V \times V$ is a set of edges. Every edge $(i, j) \in E$ corresponds to the comunication link between processors i and j. The underlying parallel program is assumed to comprise a large number of independent processes which are the basic units of workload. One or more processes may be running in a processor at any time. The total workload is assumed to be fixed, i.e., no processes are created or killed, during the execution of the load balancing procedure. We quantify the workload distribution by a vector $W = (w_1, w_2, \ldots, w_N)^T$, where w_i denotes the workload of processor i which is in terms of the number of residing processes. We assume the number of processes is large so that the workload of a node is an infinitely divisible real quantity. It is not difficult to see that without this assumption (resulting in the "integer version" which will be discussed in Section 5) our results still hold. The load balancing task is to redistribute the system workload such that each node would end up with the same $W = \sum w_i/N$, $i = 1, 2, \ldots, N$.

Note that the load balancing problem resembles in certain ways another distributed decision problem, the agreement problem [2]. This latter problem requires the nodes of a system to reach an agreement on a common scalar value, such as the average, the maximum, or the minimum, based on their own values. The load balancing problem, however, requires the nodes not only to reach an agreement on the average load, but also to adjust their workload accordingly in an automatic and efficient manner.

The computation model just described might seem restrictive, yet it is applicable to many practical problems. The static workload assumption is valid in cases where the computation is temporarily suspended for load bal-

ancing and resumed after load balancing. This is why tuning the efficiency of the load balancing is of top priority. Examples of such cases can be easily found in dynamic remapping of multiphase data-parallel computations [12, 11]. The assumption of independent processes is also reasonable in this kind of computation because the processing nodes would alternate between execution and communication in each phase and the performance is dominated by the execution time. The practical appilcability of iterative load balancing will be demonstrated through implementation of data-parallel computations in Section 5.

The GDE method is based on edge coloring of G. The edges of G are supposed to be colored beforehand with the least number of colors (κ, say), and no two adjoining edges are assigned the same color. We index the colors with integers from 1 to κ, and represent the κ-color graph as $G_\kappa = (V, E_\kappa)$, of which E_κ is a set of 3-tuples of the form $(i, j; c)$, $(i, j; c) \in E_\kappa$ if and only if c is the chromatic index of the edge $(i, j) \in E$. Figure 1 shows examples of color graphs of rings and chains. The numbers in parentheses are the assigned chromatic indices.

With the GDE method, a processor would exchange load with each of its neighbors in turn according to the order in which chromatic indices are considered in each sweep of the iterative process. A sweep corresponds to going through all the chromatic indices one—this is why we need to use the least number of colors. For processor i, the exchange of workload with a neighbor j is executed as

$$w_i = (1 - \lambda)w_i + \lambda w_j, \tag{1}$$

where w_i and w_j are the current workloads of processors i and j, respectively, and λ is the exchange parameter chosen beforehand for the given network. Note that when $\lambda = 1/2$, the GDE method is reduced to the DE method in [5, 8, 15].

For a κ-color graph, a sweep of the GDE algorithm comprises κ steps which will cover all the neighbors of every node for workload exchange. Let t be the sweep index, $t = 0, 1, 2, 3, \ldots$, and $w_i^t (1 \le i \le N)$ be the local workload of processor i at sweep t. Then the overall workload distribution at sweep t is denoted by the vector $W^t = (w_1^t, w_2^t, \ldots, w_N^t)^T$. Suppose W^0 is the initial workload distribution. Then the change of the workload distribution in the system at sweep t can be modeled by the equation

$$W^{t+1} = M(\lambda)W^t, \tag{2}$$

where $M(\lambda)$ is called the *GDE matrix* of the κ-color graph G_κ, and $M(\lambda) = M_\kappa(\lambda) \times M_{\kappa-1}(\lambda) \times \cdots \times M_1(\lambda)$. Each $M_c(\lambda)$ $(1 \le c \le \kappa)$ reflects the change of the workload distribution of the system at *step* c of sweep t.

To make dynamic load balancing work, there are two main issues. One is the termination condition; the other is efficiency, i.e., the time needed for the system to arrive at its terminated, load balanced state. The former concerns the convergence of the sequence $\{M^t(\lambda)\}$, and the latter is reflected by the asymptotic convergence rate, $R_\infty(M(\lambda))$.

Given the fact that $M(\lambda)$ is nonnegative and doubly stochastic when $0 \le \lambda \le 1$ and primitive when $0 < \lambda < 1$, it can then be shown that $0 < \lambda < 1$ is a necessary and sufficient condition for the termination of the GDE method from any initial load distribution (Theorem 3.1 of [21]).

Regarding the convergence rate, the eigenvalues of $M(\lambda)$ play a fundamental role. Let $\mu_j(M(\lambda))$ $(1 \le j \le N)$ be the eigenvalues of $M(\lambda)$, and $\rho(M(\lambda))$ and $\gamma(M(\lambda))$ be the dominant and subdominant eigenvalues, respectively, of $M(\lambda)$ in modulus. Since $\rho(M(\lambda))$ is unique and equal to 1, it follows that $R_\infty(M(\lambda)) = -\ln \gamma(M(\lambda))$. $\gamma(M(\lambda))$ is also referred to as the *convergence factor* of the GDE method in the corresponding κ-color graph. Thus, the task here is to choose a λ so that $\gamma(M(\lambda))$ is as close to 0 as possible, i.e., $R_\infty(M(\lambda))$ as large as possible.

To find the minimum $\gamma(M(\lambda))$, we have to first construct the GDE matrix $M(\lambda)$. The representation of each element m_{ij} in $M(\lambda)$ is based on the concept of *color path*. A color path of length l from vertex i to vertex j is a sequence of edges of the form

$$(i = i_0, i_1; c_1), \quad (i_1, i_2; c_2), \ldots, (i_{l-1}, i_l = j; c_l),$$

where all intermediate vertices $i_s (1 \le s \le l - 1)$ are distinct and $\kappa \ge c_1 > c_2 > \cdots > c_l \ge 1$. It indicates that processor i will receive some workload from processor j along the path in an iteration sweep. Two color paths from i to j in G_κ are said to be distinct if their intermediate vertices do not coincide at all. All the distinct color paths from i to j comprise a set $\mathcal{P}_{ij}$. Since the computation formulas for the elements of $M(\lambda)$ will be referred to frequently in the remainder of this paper, we reproduce below the lemma that defines them. Examples of GDE matrices can be found in later sections of this paper.

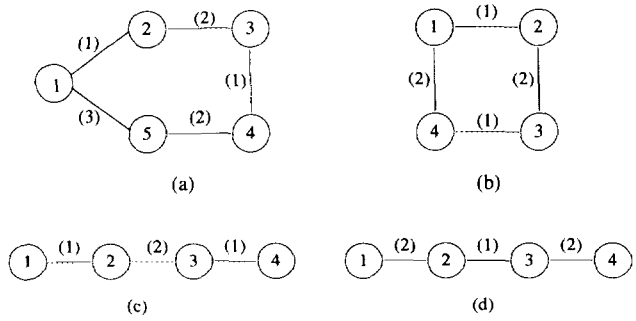


FIG. 1. Examples of colored graphs.

LEMMA 2.1. (Lemma 3.2 of [21]). *Let $M(\lambda)$ be the* GDE *matrix of a κ-color graph G_κ. If $0 < \lambda < 1$, then for*

$1 \le i, j \le N$

$$m_{ij} = \sum_{p \in \mathcal{P}_{ij}} ((1-\lambda)^{r_p} \lambda^{l_p}) \quad i \ne j$$
$$m_{ii} = (1-\lambda)^{\delta(i)} + \sum_{p \in \mathcal{P}_{ii}} ((1-\lambda)^{r_p} \lambda^{l_p}),$$

where l_p is the length of the color path $p \in \mathcal{P}_{ij}$ of the form

$$(i = i_0, i_1; c_1), \quad (i_1, i_2; c_2), \ \ldots, \ (i_{l_p - 1}, i_{l_p} = j; c_{l_p})$$

and $\delta(i)$ is the degree of vertex i, and $r_p = \sum_{s=0}^{l_p} n_s$, where n_0 is the number of incident edges of i whose chromatic index is larger than c_1, n_{l_p} is the number of incident edges of j whose color number is smaller than c_{l_p}, and n_s $(1 \le s \le l_p - 1)$ is the number of incident edges of i_s whose chromatic index is larger than c_{s+1} and smaller than c_s.

Our objective is to determine the optimal exchange parameter λ_{opt} for a given GDE matrix, which would minimize $\gamma(M(\lambda))$ and maximize the convergence rate. For arbitrary networks, this is an open problem in matrix theory [17]. For some networks with a regular topology, however, it is possible to analyze exactly the effect of λ on the convergence rate as well as to derive the optimal exchange parameter. In [21], we proved that $\lambda = 1/2$ (equal splitting of workload between a pair of nearest neighbors), which was "built-in" in the DE method, is indeed the optimal choice for certain structures (the hypercube, for example). For other structures, such as the k-ary n-cube, however, $\lambda = 1/2$ is not the optimal choice, as we will show in the next section.

3. ANALYSIS OF THE k-ARY n-CUBE AND VARIANTS

We begin with the ring structure, i.e., a k-ary 1-cube, and then generalize it to the n-dimensional $k_1 \times k_2 \times \cdots k_n$ torus. The analysis of the torus depends on the analysis of the ring, as the former can be treated as an assembly of rings. The main result for the k-ary n-cube follows trivially from the results for the torus.

The modeling tool we use for the analysis is a special kind of matrices called *block circulant matrices*. It happens that the GDE matrices of the "even" cases of the above structures—even number of nodes in every dimension—are block circulant matrices. We concentrate on these even cases for the remainder of this paper and make the remark here that the results for the even cases should be applicable (approximately) to the noneven cases. We give simulation results and a simple argument in Section 5 to support this. Nevertheless, the analysis of the noneven cases should still be an interesting theoretical problem to tackle.

For the subsequent analysis, we need to make use of the following two lemmas concerning block circulant matrices and *direct products* of matrices. We omit the proofs, which can be easily derived based on the theory of circulant matrices [7].

Let $A_1, A_2, \ldots, A_m$ be square matrices of order r. Then a *block circulant matrix* is a matrix of the form

$$\Phi(A_1, A_2, \ldots, A_m) = \begin{pmatrix} A_1 & A_2 & \cdots & \cdots & A_m \\ A_m & A_1 & \cdots & \cdots & A_{m-1} \\ & \cdots & & \cdots & \\ A_2 & A_3 & \cdots & \cdots & A_1 \end{pmatrix}.$$

If $r = 1$, a block circulant matrix degenerates to a circulant matrix.

LEMMA 3.1. *Let matrix $A = \Phi(A_1, A_2, \ldots, A_m)$. Then the eigenvalues of the matrix A are those of matrices $A_1 + \omega^j A_2 + \cdots + \omega^{j(m-1)} A_m$, $j = 0, 1, \ldots, m-1$, where $\omega^j = \cos(2\pi j/m) + i \sin(2\pi j/m)$, $i = \sqrt{-1}$. In particular, if matrix $A = \Phi(A_1, A_2)$, then the eigenvalues of A are those of $A_1 + A_2$, together with those of $A_1 - A_2$.*

Let A and B be square matrices of order m and n, respectively. Then the *direct product* of A and B is a matrix of orders $m \times n$ defined by

$$A \otimes B = \begin{pmatrix} a_{0,0}B & a_{0,1}B & \cdots & a_{0,m-1}B \\ a_{1,0}B & a_{1,1}B & \cdots & a_{1,m-1}B \\ & \cdots & & \cdots \\ a_{m-1,0}B & a_{m-1,1}B & \cdots & a_{m-1,m-1}B \end{pmatrix}.$$

LEMMA 3.2. *Let A, B be square matrices of order m and n with eigenvalues $\mu_i(A)$ and $\mu_j(B)$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$. Then the eigenvalues of $A \otimes B$ are $\mu_i(A) \times \mu_j(B)$.*

For simplicity of notation, we let

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \end{pmatrix} = \begin{pmatrix} \lambda(1-\lambda) & (1-\lambda)^2 \\ \lambda^2 & \lambda(1-\lambda) \end{pmatrix},$$

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \end{pmatrix} = \begin{pmatrix} \lambda(1-\lambda) & \lambda^2 \\ (1-\lambda)^2 & \lambda(1-\lambda) \end{pmatrix},$$

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{pmatrix} = \begin{pmatrix} 1-\lambda & \lambda \\ \lambda & 1-\lambda \end{pmatrix}.$$

3.1. The Ring Structure

The ring is a k-ary 1-cube structure whose nodes we label 1 through k. An even ring can be colored with 2 colors, as in Fig. 1b. This coloring is unique i.e., there is only way of coloring the edges without respect to the permutation of labels. The GDE matrix of an even ring is in block circulant form, as follows.

LEMMA 3.3. *Let GR_k be a color ring of even order k and MR_k be its* GDE *matrix. Then MR_k is a matrix of the form*

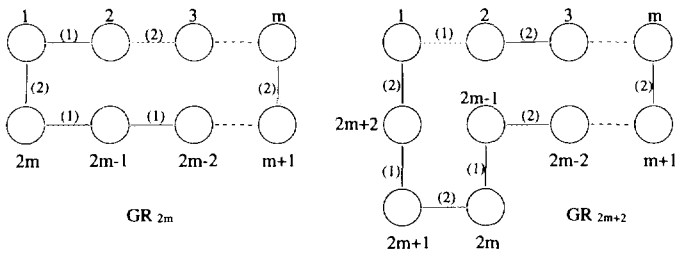


FIG. 2. An expansion of the color ring GR_{2m}.

$$\begin{pmatrix} \mathbf{V}_2 & & & \mathbf{U}_2 \\ \mathbf{U} & \mathbf{V} & & \\ & \mathbf{U} & \mathbf{V} & \\ & & \ddots & \ddots \\ & & \mathbf{U} & \mathbf{V} \\ \mathbf{V}_1 & & & \mathbf{U}_1 \end{pmatrix}_{k\times k} \tag{3}$$

Proof. The proof is by induction on the order of GR_k. First, it is easy to verify that MR_4 is in the form of (3). We need to show that if the GDE matrix MR_{2m} *of* GR_{2m}, $m \geq 2$, is in the form of (3), then the GDE matrix MR_{2m+2} of GR_{2m+2} is necessarily in that form as well. Let us view GR_{2m+2} as an expansion of GR_{2m} with two extra vertices, as illustrated in Fig. 2. The vertex labeled $2m$ in GR_{2m} is relabeled $2m + 2$, and the newly added vertices are labeled $2m$ and $2m + 1$. Then, according to the computation formulas in Lemma 2.1, we obtain the following.

For $j = 2m + 1, 2m + 2$,

$$(MR_{2m+2})_{1,j} = (MR_{2m})_{1,j-2},$$
$$(MR_{2m+2})_{2m+2,j} = (MR_{2m})_{2m,j-2},$$
$$(MR_{2m+2})_{1,j-2} = 0;$$

for $j = 1, 2$,

$$(MR_{2m+2})_{2m+2,j} = (MR_{2m})_{2m,j},$$
$$(MR_{2m+2})_{2m,j} = 0;$$

for $i = 2m, 2m + 1$; $2m - 1 \leq j \leq 2m + 2$,

$$(MR_{2m+2})_{i,j} = (MR_{2m})_{i-2,j-2}.$$

Hence, MR_{2m+2} and, therefore, MR_k are in the form of (3). ■

As an example, the GDE matrix of the ring of order 8, MR_8, is as in Fig. 3.

Given this particular structure of the matrix MR_k, we can then derive the optimal exchange parameter and determine the effect of the ring order k on the convergence rate.

THEOREM 3.1. *Let GR_k be a color ring of even order k, MR_k be the* GDE *matrix of GR_k, and $k = 2m$. Then the optimal exchange parameter $\lambda_{opt}(MR_k)$ is equal to $1/(1 + \sin(\pi/m))$, and for a given λ, $R_\infty(MR_k) \leq R_\infty(MR_{k-2})$.*

Proof. Consider the particular form of MR_k as shown in Lemma 3.3. It is easy to see that MR_k can be represented by a block circulant matrix $\Phi(A_1, A_2, 0, \ldots, 0, A_m)$, where

$$A_1 = \begin{pmatrix} (1-\lambda)^2 & \lambda(1-\lambda) \\ \lambda(1-\lambda) & (1-\lambda)^2 \end{pmatrix},$$
$$A_2 = \begin{pmatrix} 0 & 0 \\ \lambda(1-\lambda) & \lambda^2 \end{pmatrix},$$
$$A_m = \begin{pmatrix} \lambda^2 & \lambda(1-\lambda) \\ 0 & 0 \end{pmatrix}.$$

And for $j = 0, 1, \ldots, m - 1$,

$$A_1 + \omega^j A_2 + \omega^{j(m-1)} A_m = \begin{pmatrix} (1-\lambda)^2 + \omega^{m-j}\lambda^2 & \lambda(1-\lambda) + \omega^{m-j}\lambda(1-\lambda) \\ \lambda(1-\lambda) + \omega^j\lambda(1-\lambda) & (1-\lambda)^2 + \omega^j\lambda^2 \end{pmatrix}$$

because $\omega^{j(m-1)} = \omega^{m-j}$. From Lemma 3.1, the eigenvalues of the matrix MR_k are the roots of the equation

$$\mu^2 - 2((1-\lambda)^2 + \lambda^2\cos(2\pi j/m))\,\mu + (1-2\lambda)^2 = 0.$$

$$\begin{pmatrix} (1-\lambda)^2 & \lambda(1-\lambda) & 0 & 0 & 0 & 0 & \lambda^2 & \lambda(1-\lambda) \\ \lambda(1-\lambda) & (1-\lambda)^2 & \lambda(1-\lambda) & \lambda^2 & 0 & 0 & 0 & 0 \\ \lambda^2 & \lambda(1-\lambda) & (1-\lambda)^2 & \lambda(1-\lambda) & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda(1-\lambda) & (1-\lambda)^2 & \lambda(1-\lambda) & \lambda^2 & 0 & 0 \\ 0 & 0 & \lambda^2 & \lambda(1-\lambda) & (1-\lambda)^2 & \lambda(1-\lambda) & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda(1-\lambda) & (1-\lambda)^2 & \lambda(1-\lambda) & \lambda^2 \\ 0 & 0 & 0 & 0 & \lambda^2 & \lambda(1-\lambda) & (1-\lambda)^2 & \lambda(1-\lambda) \\ \lambda(1-\lambda) & \lambda^2 & 0 & 0 & 0 & 0 & \lambda(1-\lambda) & (1-\lambda)^2 \end{pmatrix}$$

FIG. 3. The GDE matrix of the ring of order 8, MR_8.

That is,

$$\mu = (1 - \lambda)^2 + \lambda^2 \cos\left(\frac{2\pi j}{m}\right) \pm \lambda \sqrt{(1 + \cos(2\pi j/m))((1 + \cos(2\pi j/m))\lambda^2 - 4\lambda + 2)},$$

where $j = 0, 1, \ldots, m - 1$.

Clearly, the dominant eigenvalue of MR_k in modulus, $\rho(MR_k)$, is equal to 1, and when $k \neq 4$, the subdominant eigenvalue of MR_k in modulus, $\gamma(MR_k)$, is equal to

$$\begin{cases} 2\lambda - 1 & \text{if } \dfrac{2 - \sqrt{2(1 - e)}}{1 + e} \leq \lambda < 1 \\ (1 - \lambda)^2 + \lambda^2 e + \lambda \sqrt{(1 + e)((1 + e)\lambda^2 - 4\lambda + 2)} & \text{if } 0 < \lambda \leq \dfrac{2 - \sqrt{2(1 - e)}}{1 + e}, \end{cases}$$

where $e = \cos(2\pi/m)$. Therefore, for a given ring of order $n = 2m$, $n \neq 4$, its optimal exchange parameter is as follows:

$$\lambda_{\text{opt}}(MR_k) = \frac{2 - \sqrt{2(1 - \cos(2\pi/m))}}{1 + \cos(2\pi/m)} = \frac{1}{1 + \sin(\pi/m)}.$$

When $k = 4$, we have binary 2-cube (two-dimensional hypercube) for which $\lambda_{\text{opt}}(MR_4) = 1/2$ and $\gamma(MR_4) = 0$, which is in agreement with previous results for the hypercube [5]. Moreover, $\gamma(MR_k)$ increases with m for a given λ. Hence, $R_\infty(MR_k)$ decreases as m increases. That is, for a given λ, $R_\infty(MR_k) \leq R_\infty(MR_{k-2})$. ■

The above theorem says that for a given λ, the more vertices an even ring has, the slower its convergence rate. It also gives the formula for the optimal λ for any even ring, which can be used in practice to compute the exact optimal value for the exchange parameter for a given ring. For example,

$$\lambda_{\text{opt}}(MR_{16}) = 2/(2 + \sqrt{2 - \sqrt{2}}) \approx 0.723.$$

3.2. The Torus Structure

The k-ary n-cube is a special case of the n-dimensional $k_1 \times k_2 \times \cdots \times k_n$ torus. In this case, $k_1 = k_2 = \cdots = k_n = k$. The general case appears to be more interesting in terms of its analysis. We first consider the two-dimensional $k_1 \times k_2$ torus with an even number of nodes in both dimensions. It can be viewed as a collection of vertical and horizontal even rings (see Fig. 4), and hence results in the previous section for the ring can be applied to the analysis here. To handle the degenerate case of k_1 or k_2 equal to 2, we use the GDE matrix for a *chain* of order 2. The reason for this is that a ring of two nodes is equivalent to a chain of two nodes as far as the dimension exchange operator is concerned. Therefore, we let

$$MR_2 = \begin{pmatrix} 1 - \lambda & \lambda \\ \lambda & 1 - \lambda \end{pmatrix}.$$

Figure 4 shows the two common ways of labeling the nodes of the torus: row-major and snakelike row-major labeling. As the spectrum of eigenvalues of the GDE matrix of a network is invariant for any permutation of the node labels, we arbitrarily choose the snakelike rowmajor labeling. Similarly, there are two ways of coloring the edges: row-major and column-major coloring, as shown in the figure. We arbitrarily assume that the torus is colored in the row-major way. Note that, with this coloring, all horizontal edges are smaller than the vertical edges in chromatic indices. We will soon see that, because of the particular structure of the GDE matrix as revealed by the following lemma, these two colorings have the same effect on the convergence rate.

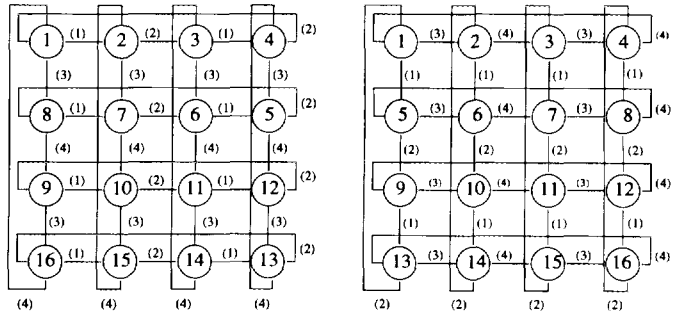
LEMMA 3.4. *Let MT_{k_1,k_2} be the* GDE *matrix of a $k_1 \times k_2$ even color torus GT_{k_1,k_2}. Then $MT_{k_1,k_2} = MR_{k_2} \otimes MR_{k_1}$.*

Proof. Given an ordered pair of vertices $\langle i, j \rangle$ in GT_{k_1,k_2},

(i) if both i and j are in the same horizontal ring, i.e., $\lfloor i/k_1 \rfloor = \lfloor j/k_1 \rfloor$, then because of the coloring we use in which all horizontal edges are smaller than the vertical edges in chromatic indices,

$$(MT_{k_1,k_2})_{i,j} = \begin{cases} (1 - \lambda) \times (MR_{k_1})_{i \bmod k_1,\, j \bmod k_1} & \text{if } k_2 = 2 \\ (1 - \lambda)^2 \times (MR_{k_1})_{i \bmod k_1,\, j \bmod k_1} & \text{otherwise} \end{cases}$$
$$= (MR_{k_2})_{\lfloor i/k_1 \rfloor, \lfloor j/k_1 \rfloor} \times (MR_{k_1})_{i \bmod k_1,\, j \bmod k_1};$$

(ii) if i and j are in different horizontal rings, i.e., $\lfloor i/k_1 \rfloor \neq \lfloor j/k_1 \rfloor$, then there is a color path from i to j, say p_{ij}, if and only if there exists a vertex i_1 such that $i_1 \bmod k_1 = i$



snake-like row-major node-labeling
row-major edge-coloring

(a)

row-major node-labeling
column-major edge-coloring

(b)

FIG. 4. Color tori of 4 × 4.

mod k_1, $\lfloor i_1/k_1 \rfloor = \lfloor j/k_1 \rfloor$, and there exists two color paths $p_{i_1 j}$ and $p_{i i_1}$ from i_1 to j and from i to i_1, respectively, as illustrated in Fig. 5. Let l_1 and l_2 be the lengths of $p_{i_1 j}$ and $p_{i i_1}$, respectively; r_1 be the sum of (i) the number of incident horizontal edges of i_1 that are between, in chromatic indices, the incident edges of i_1 along the path p_{ij}, and (ii) the number of incident horizontal edges of i_1 whose chromatic index is less than that of the last edge in p_{ij}, and r_2 be the sum of (i) the number of incident vertical edges of i_1 that are between, in chromatic indices, the incident edges of i_1 along p_{ij}, and (ii) the number of incident vertical edges of i_1 whose chromatic index is larger than that of the first edge in p_{ij}. Then, according to the computation formulas in Lemma 2.1, we have

$$\begin{aligned}(MT_{k_1,k_2})_{i,j} &= \lambda^{l_1+l_2}(1-\lambda)^{r_1+r_2} \\ &= \lambda^{l_2}(1-\lambda)^{r_2}\lambda^{l_1}(1-\lambda)^{r_1} \\ &= (MR_{k_2})_{\lfloor i/k_1 \rfloor, \lfloor i_1/k_1 \rfloor} \times (MR_{k_1})_{i_1 \bmod k_1,\, j \bmod k_1} \\ &= (MR_{k_2})_{\lfloor i/k_1 \rfloor, \lfloor j/k_1 \rfloor} \times (MR_{k_1})_{i \bmod k_1,\, j \bmod k_1}.\end{aligned}$$

By referring to the definition of direct product in Lemma 3.2, the lemma is proved. ■

If instead we use column-major coloring, then all horizontal edges are larger than the vertical edges in chromatic indices. By following the above steps, however, we would find that the resulting GDE matrix is the same as MT_{k_1,k_2}. We continue to assume row-major coloring in the following discussion.

We now turn to the convergence rate of GDE in the torus and see how it is related to the convergence rate in the ring.

THEOREM 3.2. *Let MT_{k_1,k_2} be the* GDE *matrix of a $k_1 \times k_2$ even color torus GT_{k_1,k_2}. Then, the optimal exchange parameter $\lambda_{\text{opt}}(MT_{k_1,k_2})$ is equal to $\lambda_{\text{opt}}(MR_k)$, and for a given λ, $R_\infty(MT_{k_1,k_2}) = R_\infty(MR_k)$, where $k = \max\{k_1, k_2\}$.*

Proof. From Lemma 3.2, it is clear that

$$\gamma(MT_{k_1,k_2}) = \max\{\gamma(MR_{k_1}), \gamma(MR_{k_2})\}$$

because of $\rho(MR_{k_1}) = \rho(MR_{k_2}) = 1$. Moreover, from Theorem 3.1, $\gamma(MR_{k_1}) \geq \gamma(MR_{k_2})$ if and only if $k_1 \geq k_2$. Hence, $\gamma(MT_{k_1,k_2}) = \gamma(MR_k)$, where $k = \max\{k_1, k_2\}$, and $\gamma(MT_{k_1,k_2})$ is minimized when $\lambda = \lambda_{\text{opt}}(MR_k)$. In other words, $R_\infty(MT_{k_1,k_2}) = R_\infty(MR_k)$ and both MR_k and MT_{k_1,k_2} *have the same optimal exchange parameter.* ■

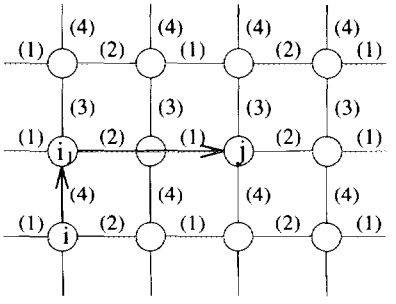


FIG. 5. Illustration of the color path from vertex i to vertex j.

As a result, one can compute the optimal exchange parameter $\lambda_{\text{opt}}(MT_{k_1,k_2})$ using the formula in Theorem 3.1. Moreover, the above theorem shows that the convergence rate in two-dimensional torus structures depends only on the larger dimension. For example, the tori $GT_{16,j}$, $j = 2, 4, \ldots, 16$, all have the same convergence rate for a given λ and share the same optimal exchange parameter

$$\lambda_{\text{opt}}(MT_{16,j}) = 2/(2 + \sqrt{2 - \sqrt{2}}) \approx 0.723.$$

The results for the two-dimensional torus shown above can be generalized to multidimensional cases. Consider a $k_1 \times k_2 \times \cdots \times k_n$ even torus and assume that this n-dimensional torus is colored in a way similar to that for the two-dimensional torus. Then, its GDE matrix can be expressed in terms of direct products of color rings.

LEMMA 3.5. *Let $MT_{k_1,k_2,\ldots,k_n}$ be the* GDE *matrix of an n-dimensional $k_1 \times k_2 \times \cdots \times k_n$ even color torus $GT_{k_1,k_2,\ldots,k_n}$. Then, $MT_{k_1,k_2,\ldots,k_n} = MR_{k_1} \otimes MR_{k_2} \otimes \cdots \otimes MR_{k_n}$.*

We omit the tedious proof here which is based on induction on the dimension n. From this lemma, the following result is immediately in order.

THEOREM 3.3. *Let $MT_{k_1,k_2,\ldots,k_n}$ be the GDE matrix of an n-dimensional $k_1 \times k_2 \times \cdots \times k_n$ even color torus $GT_{k_1,k_2,\ldots,k_n}$. Then, the optimal exchange parameter $\lambda_{\text{opt}}(MT_{k_1,k_2,\ldots,k_n})$ is equal to $\lambda_{\text{opt}}(MR_k)$, where $k = \max_{1 \leq j < n}\{k_j\}$, and for a given λ, $R_\infty(MT_{k_1,k_2,\ldots,k_n}) = R_\infty(MR_k)$.*

Since a k-ary n-cube is a special case of an n-dimensional torus, we have the following major result for the k-ary n-cube.

COROLLARY 3.1. *Let $MT_{k;n}$ be the* GDE *matrix of a color k-ary n-cube, k even. Then the optimal exchange parameter $\lambda_{\text{opt}}(MT_{k;n})$ is equal to $\lambda_{\text{opt}}(MR_k)$, and for a given λ, $R_\infty(MT_{k;n}) = R_\infty(MR_k)$.*

3.3. Summary of Theoretical Results

The last theorem and its corollary in the previous section equate the convergence rates of the ring, the torus, and the k-ary n-cube, therefore leading to this important conclusion: given a fixed number of nodes, the best way to connect them as far as GDE load balancing is concerned is as k-ary n-cube. Then, from the above analysis, we find that the smaller the value of k, the better the convergence rate; hence, the binary n-cube is the best choice, which takes exactly one sweep to balance the workload. However, as Dally points out in [6], there are

TABLE I
Comparison between the Convergence Factors of the GDE Method and the Diffusion Method in $k_1 \times k_2 \times \cdots \times k_n$

	GDE method		Diffusion method	
	λ_{opt}	$\gamma(M(\lambda_{opt}))$	α_{opt}	$\gamma(D(\alpha_{opt}))$
Torus	$\frac{1}{1+\sin(2\pi/k)}$	$\frac{2}{1+\sin(2\pi/k)} - 1$	$\frac{1}{2n+1-\cos(2\pi/k)}$	$\frac{4n}{2n+1-\cos(2\pi/k)} - 1$
Mesh	$\frac{1}{1+\sin(\pi/k)}$	$\frac{2}{1+\sin(\pi/k)} - 1$	$\frac{1}{2n}$	$1 - \frac{1}{n} + \frac{1}{n}\cos(\pi/k)$

Note. k_i is even and $k = \max\{k_i\}$, $i = 0, 1, \ldots, n$.

other practical reasons for which a k-ary n-cube with a bigger k is preferable.

The analysis technique as exemplified in the above section can also be applied to the chain and the mesh which can be viewed as variants of the ring and the torus by deleting the end-round connections. The detailed proofs for the following two theorems can be found in [19].

THEOREM 3.4. *Let GC_k be a color chain of even order k, and MC_k be its* GDE *matrix. Then the optimal exchange parameter $\lambda_{opt}(MC_k)$ is equal to $\lambda_{opt}(MR_{2k})$, and for a given λ, $R_\infty(MC_k) = R_\infty(MR_{2k})$.*

THEOREM 3.5. *Let $GM_{k_1,k_2,\ldots,k_n}$ be an n-dimensional even color mesh, and $MM_{k_1,k_2,\ldots,k_n}$ be its* GDE *matrix. Then the optimal exchange parameter $\lambda_{opt}(MM_{k_1,k_2,\ldots,k_n})$ is equal to $\lambda_{opt}(MR_k)$, and for a given λ, $R_\infty(MM_{k_1,k_2,\ldots,k_n}) = R_\infty(MC_k)$, where $k = \max_{1\le j\le n}\{k_j\}$.*

Based on these theorems, the optimal exchange parameters for even chains and meshes can be easily obtained. For example, for $j = 2, 4, 6, 8$,

$$\lambda_{opt}(MM_{8,j}) = \lambda_{opt}(MC_8) = \lambda_{opt}(MR_{16}) \approx 0.723.$$

These theorems also show that the convergence rate of a mesh depends only on its largest dimension. Now, based on these theorems and those in the previous section, we can establish the relationships between the convergence rates of the ring, the torus, the chain, and the mesh. Here is a summary of the results. Note that the results for the k-ary n-cube are implicit in the results for the torus.

(i) Suppose each k_i, $1 \le i \le n$, is even, and $k = \max\{k_i,\ 1 \le i \le n\}$. Then,

$$\lambda_{opt}(MR_{2k} = \lambda_{opt}(MT_{2k_1,2k_2,\ldots,2k_n} = \lambda_{opt}(MC_k) = \lambda_{opt}(MM_{k_1,k_2,\ldots,k_n}),$$

which is equal to $1/(1 + \sin(\pi/k))$.

(ii) For even rings and chains and a given λ, the more vertices the structure has, the slower the convergence rate is.

(iii) For tori and meshes of even order (even number of nodes in each dimension), the convergence rate depends only on the largest dimension.

(iv) The convergence rates of these four structures are related as follows.

$$R_\infty(MR_{2k}) = r_\infty(MT_{2k_1,2k_2,\ldots,2k_n}) = R_\infty(MC_k)k = R_\infty(MM_{k_1,k_2,\ldots,k_n}),$$

where each k_i, $1 \le i \le n$, is even, and $k = \max\{k_i,\ 1 \le i \le n\}$.

4. COMPARISON WITH THE DIFFUSION METHOD

We have derived the optimal exchange parameters for leading to the fastest asymptomatic convergence rate in the k-ary n-cube and its invariants. Here, we would like to compare the GDE method with another relaxation-based method, the diffusion method [3, 5, 23]. The measure of interest is still the convergence rate.

With the diffusion method, a processor would interact with all its neighbors *simultaneously* at each step. For processor i, the change of workload in a processor i is executed as

$$w_i = w_i + \sum_{j\in A(i)} \alpha(w_j - w_i), \tag{4}$$

where $A(i)$ is the set of nearest neighbors and α is the *diffusion parameter* which determines the portion of excess of workload to be diffused away. As a whole, the change of the workload distribution at step t is modeled by the equation

$$W^{t+1} = D(\alpha)W^t, \tag{5}$$

where $D(\alpha)$ is the *diffusion matrix*,[2] as given in [5].

The efficiency of the diffusion method is reflected by the asymptomatic convergence rate, $R_\infty(D(\alpha))$, which is equal to $-\ln \gamma(D(\alpha))$. $\gamma(D(\alpha))$ is the subdominant eigenvalue of $D(\alpha)$ and is also referred to as the *convergence factor* of the diffusion method. We have previously derived the optimal diffusion parameters for the k-ary n-cube and its variants [23]. Table I summarizes the optimal parameter values and their corresponding convergence

[2] The same α is used for the entire network. It is possible to use different α's for different edges of the network, as discussed in [5].

factors. For comparison, the results of the GDE method are also included. Clearly, $\gamma(M(\lambda)) < \gamma(D(\alpha))$ in both torus and mesh structures.

In multicomputers, there are two basic communication models. One is the serial communication model which restricts a node to communicating with at most one nearest neighbor at a time; the other is the parallel communication model which allows a node to communicate with all its nearest neighbors simultaneously. Clearly, the serial communication model favors the dimension exchange method and the parallel model favors the diffusion method. Recall that the GDE matrix $M(\lambda)$ reflects a complete sweep, i.e., κ consecutive iteration steps, each of which involves an I/O communication at a node. On the other hand, the diffusion matrix $D(\alpha)$ reflects a single iteration step which involves $\Delta(G)$ I/O communications at a node, where $\Delta(G)$ is the maximum degree of the nodes. Hence, $R_\infty(M(\lambda_{opt})) > R_\infty(D(\alpha_{opt}))$ in the serial communication model.

In the parallel communication model, an n-dimensional torus or mesh can be colored by $2n$ colors, and hence a dimension exchange sweep would take as much time as that for $2n$ diffusion steps; these $2n$ diffusion steps in fact correspond to the matrix D^{2n}. We should therefore compare $\gamma(D^{2n}(\alpha_{opt}))$ with $\gamma(M(\lambda_{opt}))$. Since the diffusion matrix is symmetric, it follows that $\gamma(D^{2n}(\alpha_{opt})) = \gamma^{2n}(D(\alpha_{opt}))$. Thus, in the case of the torus,

$$\begin{aligned}\gamma(D^{2n}(\alpha_{opt})) &= \left(\frac{4}{2n + 1 - \cos(2\pi/k)} - 1\right)^{2n} \\ &\geq \left(\frac{4}{3 - \cos(2\pi/k)} - 1\right)^{2} \\ &> \gamma(M(\lambda_{opt})),\end{aligned}$$

and in the case of the mesh,

$$\begin{aligned}\gamma(D^{2n}(\alpha_{opt})) &= \left(1 - \frac{1}{n} + \frac{1}{n}\cos(\pi/k)\right)^{2n} \\ &\geq (\cos(\pi/k))^2 \\ &> \gamma(M(\lambda_{opt})).\end{aligned}$$

We have thus proved the following, which is valid for both the serial and the parallel communication models.

THEOREM 4.1. *The* GDE *method converges asymptotically faster than the diffusion method when applied to the k-ary n-cube and its variants.*

In [5], Cybenko compared the efficiencies of these two methods when applied to the binary n-cube structure, and revealed the superiority of the dimension exchange method in both communication models. The above theorem extends his result to the family of k-ary n-cube structures.

5. SIMULATION AND PRACTICAL IMPLEMENTATIONS

For practical applications, it will be of considerable value if the number of iteration sweeps required by the GDE procedure to balance the system's load can be obtained or estimated. To this end, we conducted statistical simulation experiments on a number of test cases. In addition to giving us information on the number of iteration sweeps, the experiments reveal in measurable terms the efficiency gains due to the optimal exchange parameters. The simulation results also confirmed our theoretical results concerning the equivalence of the various convergence rates and the optimality of the derived parameter values.

In the theoretical analysis, we represent the workload of a processor by a real number, which is reasonable under the assumption of very fine grain parallelism as exhibited by the computation. To cover medium- and large-grain parallelisms which are more realistic and more common in practical parallel computing environments, one can treat the workloads of the processors more conveniently as nonnegative integers, as is done in [8]. For example, in the WaTor simulation experiment which we will discuss shortly, the workload of a processor is an integer which corresponds to the number of fishes in the strip of the ocean for which the processor is responsible. We used the integer version of the GDE method in our simulation experiments. All we need to do is to modify the exchange operator of Eq. (1) in Section 2. During exchange with a neighbor j, processor i would update its workload according to the revised formula

$$w_i = \begin{cases} \lceil (1 - \lambda)w_i + \lambda w_j \rceil & \text{if } w_i \geq w_j \\ \lfloor (1 - \lambda)w_i + \lambda w_j \rfloor & \text{otherwise.} \end{cases} \tag{6}$$

As discussed in [8], the integer version of the original DE method (i.e., $\lambda = 1/2$) is just a pertubation of its real counterpart and will converge to a nearly balanced state. Applying the pertubation theory to the real version of our GDE method verbatim, we can come to a similar conclusion.

Because of the use of integer workloads, we allow the load balancing procedure to end with a variance of some threshold value (in workload units) between neighboring processors. This threshold value can be tuned to satisfactory performance of the procedure in practice, as illustrated in [9]. In all our simulation experiments, this value is set to *one* workload unit which is closest to total balancing. Then, it is clear that $0.5 \leq \lambda < 1$ because a pair of neighboring processors with a variance of more than one workload unit would not balance their workloads any more when $\lambda < 0.5$. Since the termination condition of a processor is rather localized, we add a mechanism for global termination detection to the simulation [22]. We exclude the rather small delay for termination detection using this mechanism detection. We exclude the rather small delay for termination detection using this mecha-

nism in our simulation results, and so the number of sweeps reported below reflects purely the efficiency of the GDE method. This termination detection might not be necessary if we can set a limit on the number of sweeps for a given structure beforehand. In fact, our simulation results below can help the users of this method to set such a limit.

5.1. Number of Iteration Sweeps

The number of iteration sweeps, denoted by NS, is expected to depend upon such factors as the exchange parameter, the initial workload distribution, the topology, and the size of the underlying system structure. The initial workload distribution is a random vector, each element of which is drawn independently from an identical uniform distribution in $[0, 2b]$, where b is a prescribed bound. The mean workload a processor gets (i.e., the expected workload) is thus equal to b. The amount of workload a processor gets is determined by the distribution mean. Table II displays the expected numbers of sweeps generated by the experiments for the structures of ring 16, chain 8, 16-ary 2-cube (i.e., torus 16×16), and mesh 8×4. The initial workload distribution has a mean of 128 units per processor, and λ varies from 0.5 to 0.9 in steps of 0.05. Each data point is the average of 100 runs, each using a different random initial load distribution. The second column in the table shows the convergence factors, $\gamma(M(\lambda))$, of the GDE matrices of the various cases. From the table, it is clear that the expected number of sweeps in each case for different values of λ rises and drops with the value of $\gamma(M(\lambda))$, and that the optimal exchange parameter λ_{opt} of each case is not equal to 0.5, but somewhere between 0.7 and 0.8, which is in agreement with the theoretical result of $\lambda_{opt} = 0.723$. It also appears that the absolute values of the expected number of sweeps for the various structures are very close to each other, especially when near the optimum point, which is in line with our theoretical results on the equivalence of the convergence rates.

Furthermore, it is most encouraging to see that the optimal sweep numbers are rather small—in the neighborhood of 8 sweeps (even for a 256-processor 16-ary 2-cube!). We also tried different numbers of processors for each kind of topology. The results for chains of up to 128 processors are depicted in Fig. 6. The figure shows that the optimal number of sweeps is linearly proportional to the number of processors for the chain, and hence to the dimension order k of the k-ary n-cube structure. This really puts forth the GDE method as a practical method for load balancing in real multicomputers.

Notice that the convergence rates in the theoretical analysis are in terms of sweeps over time. A sweep of the GDE method may involve different numbers of nearest neighbor communications in various structures. In a k-ary n-cube (k even), a sweep comprises $2n$ communication steps (when $n < 2$) or n steps (when $n = 2$). Thus, for a given number of processors, a higher dimensional k-ary n-cube, even though it takes fewer sweeps to balance the load, requires more communication steps within a sweep in reality. However, from Fig. 6, we point out that the minimal number of sweeps necessary for convergence would decrease at a logarithmic rate with the increase in the number of dimensions; this is because the dimension order decreases at the same rate as the increase in the number of dimensions for a given number of processors. As an example, consider a cluster of 4096 processors, which can be organized as a structure of 64-ary 2-cube, 16-ary 3-cube, 8-ary 4-cube, or 2-ary 12-cube. The minimal sweep numbers for these structures are about 35, 8, 4, and 1 sweep, respectively. Since the number of communication steps within a sweep would only double with every added dimension, it is justified to maintain that the GDE method is most effective in high-dimensional k-ary n-cubes (in particular, the binary n-cube and the 4-ary n-cube).

5.2. The Noneven Cases

In addition to the even cases, we also simulated a few noneven cases which the theoretical analysis has not

TABLE II

Expected Number of Sweeps $E(NS)$ for Ring 16, Chain 8, 16-ary 2-cube, and Mesh 8×4

λ	$\gamma(M(\lambda))$	Ring 16	16-ary 2-cube	Chain 8	Mesh 8×4
0.50	0.8536	21.33	16.69	19.97	15.78
0.55	0.8206	20.30	16.22	19.08	15.05
0.60	0.7777	16.79	13.91	15.87	12.61
0.65	0.7170	15.17	12.95	14.28	11.37
0.70	0.6112	10.76	9.19	10.22	8.70
0.75	0.5000	8.55	7.69	8.32	7.67
0.80	0.6000	9.68	7.73	9.44	8.27
0.85	0.7000	11.63	9.14	11.54	10.14
0.90	0.8000	15.88	11.72	15.86	13.56
0.95	0.9000	25.42	17.31	25.56	20.80
0.723	0.4465	9.82	8.58	9.19	8.25

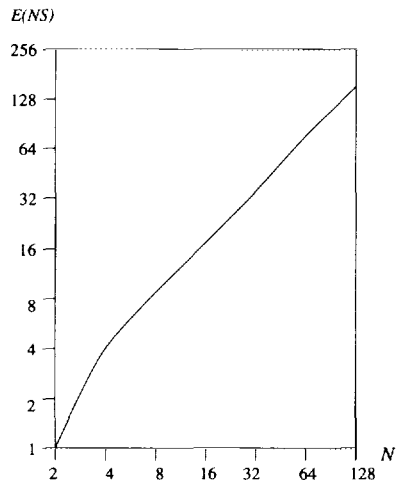


FIG. 6. Expected number of sweeps $E(NS)$ using λ_{opt} for chains of various sizes.

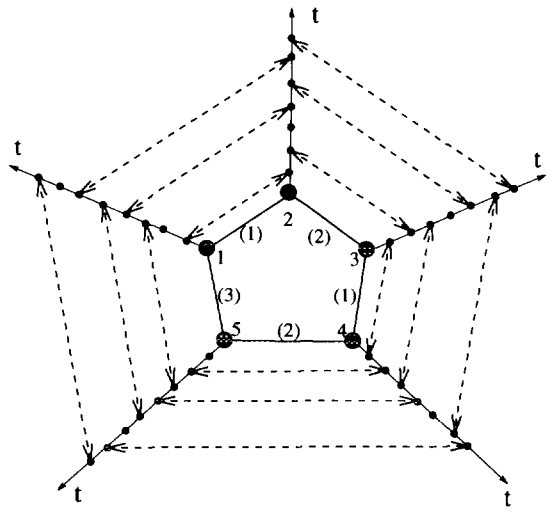


FIG. 7. Communication pattern of the GDE method in a ring of 5 nodes.

been able to deal with because of their structural differences from the even cases. We approximated their optimal exchange parameters using the formulas for the even cases. Table III summarizes the simulation results. The numbers in parentheses in the bottom row of the table are the approximated optimal parameter and the resulting number of sweeps, respectively. In line with our remarks before, the odd or noneven cases do not behave differently from the even cases in terms of their convergence pattern and the optimal values for the exchange parameter, which can be seen by comparing Table III with Table II. Hence, it is reasonable to conclude that the results for the even cases can be applied, as a close approximation, to the noneven cases.

Note that an odd ring has to be colored using three colors, as in Fig. 1(a). That is, a sweep of the GDE method in an odd ring comprises three communication steps. As only two processors are involved in workload exchange in the third communication step, it seems that much communication bandwidth may be wasted. However, a close examination of the communication pattern of GDE balancing in the odd ring, as shown in Fig. 7, would reveal that the third chromatic index would account for only a small fraction of the total communication overhead incurred in the balancing process. In the figure, the five large dots in the centre represent processors that are connected as a ring. We attach a time axis t to each processor for easy viewing. Each dashed double arrow represents a communication step between a pair of nearest neighbors for the exchange of their workloads. At time $t = 1$, all processors except the fifth are involved in communication. Then, the first processor becomes idle at time $t = 2$. While the first and fifth processors are busy executing the third communication step at time $t = 3$, the third and fourth processors are already executing the first communication of the next sweep. At this time, only the second processor is in the idle state. Continuing, we see that the GDE procedure finishes two complete iteration sweeps in five communication steps. In other words, GDE balancing in an odd ring of five nodes costs only one extra communication step more than an even ring of comparable size for two iteration sweeps. This example can

TABLE III

Expected Number of Sweeps $E(NS)$ for Ring 15, Chain 7, Torus 16 × 5, and Mesh 8 × 5

λ	Ring 15	Torus 16 × 5	Chain 7	Mesh 8 × 5
0.50	20.20	16.11	15.91	15.45
0.55	19.24	15.68	14.98	14.83
0.60	15.74	13.28	12.24	12.63
0.65	14.23	12.37	10.96	11.42
0.70	10.05	9.21	8.10	8.60
0.75	8.26	7.95	7.45	7.54
0.80	9.88	8.23	9.29	8.10
0.85	11.76	10.11	11.43	10.01
0.90	16.16	13.31	15.79	13.29
0.95	25.50	19.75	25.36	20.11
	(0.711, 9.75)	(0.723, 8.70)	(0.697, 8.23)	(0.723, 8.05)

TABLE IV
Improvements (%) for Rings of Various Sizes and Workloads

Size	Mean workload per processor			
	10	100	1000	10,000
8	3.16	20.83	29.67	37.23
16	19.21	51.79	61.97	66.49
32	12.75	67.81	79.14	84.20
64	6.50	72.80	88.43	90.80
128	9.13	72.41	92.19	94.93

TABLE VI
Improvements (%) for Chains of Various Sizes and Workloads

Size	Mean workload per processor			
	10	100	1000	10,000
8	10.68	48.20	61.88	66.33
16	3.18	65.55	79.52	82.54
32	2.00	71.25	86.93	90.40
64	1.98	69.15	91.27	94.61
128	8.38	62.55	92.22	96.95

be generalized to an odd ring of arbitrary size. In general, the GDE balancing in an odd ring of $2k + 1$ nodes costs one communication step more than that in the even ring of comparable size for k iteration sweeps. Based on the equivalence results between the ring and the other structures in previous sections, we can conclude that there is a negligibly small difference between the efficiencies of optimal GDE balancing in noneven and even cases of these structures.

5.3. *Improvements Due to the Optimal Parameters*

It is clear from the simulation results that the optimal exchange parameter λ_{opt} yields (much) better results than the choice of $\lambda = 1/2$ which is used in the original DE method. To further examine and quantify the benefits of our GDE method over the original DE method, we define a metric for measuring improvements, denoted δ,

$$\delta = \frac{NS_h - NS_{opt}}{NS_h} \times 100\%,$$

where NS_h and NS_{opt} are the expected numbers of sweeps from using $\lambda = 1/2$ and the optimal λ_{opt}, respectively. The improvement reflects the superiority of the optimal dimension exchange method (GDE) over the original one (DE). Tables IV–VII show the results for different structures; for each structure, different sizes of the structure and different workloads per processor are considered.

These results suggest that the improvement (δ) increases as the average workload per processor increases. It is not so evident under light loads but becomes significant under heavy loads. For example, in a ring with 64 nodes, if each processor is loaded with 10 units, then $NS_h = 7.39$, $NS_{opt} = 6.91$, and $\delta = 6.5\%$; if each processor is loaded with 10,000 units, then $NS_h = 543$, $NS_{opt} = 50$, and $\delta = 90.8\%$. The simulation results also show that the improvement is proportional to the size of a structure when workload is heavy, i.e., the larger the system, the better the performance of our GDE method using optimal exchange parameters. In summary, dynamic load balancing using dimension exchange does benefit substantially from the optimal exchange parameter.

5.4. *Practical Implementations*

The simulation experiments shed light on the number of iteration sweeps, but ignored the overhead that might be incurred in actual implementation of the method. Neither did they tell us anything about the expected performance gain when the method is used in real applications. We therefore took the GDE method and implemented it as the dynamic load balancer within two data-parallel computations. It turns out that the overhead due to the periodic execution of the GDE method is very small but the gain in performance through the balancing over the versions without GDE balancing is substantial. In data-parallel computations, the computational requirement associated with each portion of a problem domain may change as the computation proceeds. To reduce the penalty of load imbalances, an effective way is to periodically "remap" (redecompose) the problem domain onto the processors; the goal of this remapping is to try to create a balanced workload across the processors for the next phase of the computation [11]. To allow ourselves to evaluate the GDE method as well as to study GDE-based remapping, we implemented two major applications in a

TABLE V
Improvements (%) for Tori of Various Sizes and Workloads

Size	Mean workload per processor			
	10	100	1000	10,000
4 × 4	0	0	0	0
8 × 8	3.43	16.36	26.22	32.83
16 × 16	11.62	47.05	57.44	63.63
32 × 32	5.01	60.39	75.97	80.58

TABLE VII
Improvements (%) for Meshes of Various Sizes and Workloads

Size	Mean workload per processor			
	10	100	1000	10,000
4 × 4	3.98	17.11	27.79	34.56
8 × 8	11.60	48.01	59.01	64.39
16 × 16	5.10	61.26	77.72	81.15
32 × 32	6.82	61.30	85.57	89.57

multicomputer (a transputer array): the WaTor simulation and the parallel thinning of images. The remapping mechanism in these applications comprises two components: the decision maker and the workload adjuster. The decision maker uses the GDE method to drive the processors into a consensus on the uniform workload distribution; then the workload adjuster carries out the actual workload redistribution according to the decisions just made. This remapping mechanism is invoked periodically. In the WaTor simulation of a 256 × 256 toroidal ocean on a 16-transputer ring-structured network, it is found that frequent remapping (once every two simulation steps) leads to a 10–20% improvement on the total simulation time (for 100 steps of simulating the ocean). In parallel thinning of a 128 × 128 image (a popular image of a man's body) on an 8-transputer chain-structured network, frequent remapping using GDE yields a performance gain of 10% on thinning time even though the test image does not favor remapping. Details of and discussion about these experiments can be found in [19]. These results have led us to believe that the GDE method with the optimal exchange parameters is a viable tool for dynamic load balancing in practical implementations of data-parallel computations.

6. CONCLUDING REMARKS

We have analyzed the GDE method for dynamic load balancing as applied to the k-ary n-cube and its variants—the ring, the torus, the chain, and the mesh. We have derived the optimal exchange parameters in closed form, which maximize the convergence rates of GDE balancing in these structures. We have shown that there exist close relationships between their convergence rates and concluded that the GDE method favors high-dimensional k-ary n-cubes for a given number of processors. We have also revealed the superiority of the GDE method to the diffusion method when both are applied to these structures. Through statistical simulation experiments, we have shown that the efficiency (in terms of number of steps to convergence) of using the optimal exchange parameters is significantly better than that of the nonoptimal cases such as the original DE method.

This paper has analyzed theoretically only the even cases of the various topologies. This is due to our using matrix partitioning and circulant matrices for the analysis. It is conceivable, however, that the odd cases would behave more or less the same as their even counterparts especially when the number of nodes is large. We found this to be true for the noneven cases we simulated. We also presented an argument that suggested that the difference between the two in terms of efficiency should be negligibly small. Nevertheless, finding a different mathematical tool to analyze the odd cases also would be an interesting theoretical pursuit. In addition, after having dealt with some of the most common regular structures, it is natural to think of arbitrary structures. Unfortunately, the derivation of the optimal exchange parameter λ for arbitrary structures requires a solution to the problem of specifying, in analytical form, the dependence of the subdominant eigenvalue in modulus of a matrix on the matrix elements. This is still open in mathematics [4].

APPENDIX: LIST OF SYMBOLS

$G = (V, E)$	system graph
$\Delta(G)$	degree of the graph G
$G_\kappa = (V, E_\kappa)$	κ-color graph; κ is the chromatic index of the graph G
$M(\lambda)$	generalized dimension exchange matrix
$D(\alpha)$	diffusion matrix
P_{ij}	set of distinct color paths from vertex i to vertex j, with typical element p_{ij}
λ	exchange parameter
λ_{opt}	optimal exchange parameter
α	diffusion parameter
α_{opt}	optimal diffusion parameter
w_i^t	workload of node p_i at time t
W^t	workload distribution at time t
$\gamma(M)$	convergence factor of the GDE method
$\gamma(D)$	convergence factor of the diffusion method
$R_\infty(M(\lambda))$	asymptotic convergence rate of the sequence $\{M^t\}$
MC_k	GDE matrix of the color chain of size n
MR_k	GDE matrix of the color ring of size n
$MM_{k_1,k_2,\ldots,k_n}$	GDE matrix of the color mesh of size $k_1 \times k_2 \times \cdots \times k_n$
$MT_{k_1,k_2,\ldots,k_n}$	GDE matrix of the color torus of size $k_1 \times k_2 \times \cdots \times k_n$
DC_k	diffusion matrix of the chain of size n
DR_k	diffusion matrix of the ring of size n
$DM_{k_1,k_2,\ldots,k_n}$	diffusion matrix of the color mesh of size $k_1 \times k_2 \times \cdots \times k_n$
$DT_{k_1,k_2,\ldots,k_n}$	diffusion matrix of the color torus of size $k_1 \times k_2 \times \cdots \times k_n$
δ	improvement due to the use of λ_{opt}

ACKNOWLEDGMENT

We thank the anonymous referees for their constructive comments.

REFERENCES

1. W. C. Athas and C. L. Seitz. Multicomputers: message-passing concurrent computers. *IEEE Computer* **21**, 8 (Aug. 1988), 9–24.
2. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
3. J. B. Boillat. Load balancing and poisson equation in a graph. *Concurrency: Practice and Experience* **2**, 4 (Dec. 1990), 289–313.
4. A. Broder and E. Shamir. On the second eigenvalue of random regular graphs. In *Proc. of 28th IEEE Foundations of Computer Science*, 1987, pp. 286–294.
5. G. Cybenko. Load balancing for distributed memory multiprocessors. *J. Parallel Distrib. Comput.* **7** (1989), 279–301.

6. W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Comput.* **39**, 6 (June 1990), 775–785.
7. P. J. Davis. *Circulant Matrices*. Wiley, New York, 1979.
8. S. H. Hosseini, B. Litow, M. Malkawi, J. Mcpherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *J. Parallel Distrib. Comput.* **10** (1990), 160–166.
9. R. Lüling and B. Monien. Load balancing for distributed branch and bound algorithm. In *Proceedings of 6th International Parallel Processing Symposium*, pp. 543–548.
10. L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer* **26** (Feb. 1993), 62–76.
11. D. M. Nicol and P. F. Reynolds. Optimal dynamic remapping of data parallel computation. *IEEE Trans. Comput.* **39**, 2 (Feb. 1990), 206–219.
12. D. M. Nicol and J. H. Saltz. Dynamic remapping of parallel computations with varying resource demands. *IEEE Trans. Comput.* **37**, 9 (Sep. 1988), 1073–1087.
13. G. Ramanathan and J. Oren. Survey of commercial parallel machines. *ACM Comput. Architecture News* **21**, 3 (June 1993), 13–33.
14. S. Ranka, Y. Won, and S. Sahni. Programming a hypercube multicomputer. *IEEE Software* **5** (September 1988), 69–77.
15. Y. Shih and J. Fier. Hypercube systems and key applications. In K. Hwang and D. Degroot (Eds.) *Parallel Processing for Supercomputers and Artificial Intelligence*. McGraw-Hill, New York, 1989, pp. 203–243.
16. G. Trew, and A. Wilson. *Past, Present and Parallel: A Survey of Available Parallel Computer Systems*. Springer-Verlag, New York/Berlin, 1991.
17. R. S. Varga. *Matrix Iterative Analysis*. Prentice–Hall, Englewood Cliffs, New Jersey, 1962.
18. M. Willebeek-LeMair and A. P. Reeves. Local vs. global strategies for dynamic load balancing. In *Proceedings of International Conference on Parallel Processing*, Vol. 1, 1990, pp. 569–570.

Received July 19, 1993; revised January 10, 1994; accepted January 14, 1994

19. C.-Z. Xu. Iterative methods for dynamic load balancing in multicomputers. Ph.D. thesis, Department of Computer Science, The University of Hong Kong, 1993.
20. C.-Z. Xu and F. C. M. Lau. Iterative dynamic load balancing in multicomputers. *J. Oper. Res. Soc.* **45**, 7 (Jul. 1994), 786–796.
21. C.-Z. Xu and F. C. M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *J. Parallel Distrib. Comput.* **16**, 4 (Dec. 1992), 385–393.
22. C.-Z. Xu and F. C. M. Lau. Termination detection for loosely synchronized computations. In *Proceedings of 4th IEEE Symposium on Parallel and Distributed Processing*, (Dec. 1992), pp. 196–203.
23. C.-Z. Xu and F. C. M. Lau. Optimal parameters for load balancing using the diffusion method in the $k$ary n-cube networks. *Inform. Process. Lett.* **47**, 5 (Sep. 1993), 181–187. (A longer version appears as Tech. Report TR-93-03, Department of Computer Science, The University of Hong Kong, Mar. 1993.)

CHENG-ZHONG XU received the B.Sc. and M.S. degrees from the Nanjing University, People's Republic of China in 1986 and 1989, respectively, and the Ph.D. degree from the University of Hong Kong in 1993, all of which are in computer science. He is a lecturer at the Department of computer Science, Shantou University, People's Republic of China. His research interests are primarily in the design and analysis of algorithms for mapping problems in parallel computers, and in the development of parallel programming environments.

FRANCIS C. M. LAU received the B.Sc. degree from Acadia University, Canada in 1979, and the M.Math. and Ph.D. degrees from the University of Waterloo, Canada in 1980 and 1986, respectively, all of which are in computer science. He is a lecturer at the Department of Computer Science, University of Hong Kong. His research interests include parallel and distributed computing, operating systems, object-oriented programming, and real-time computing.