Analysis of The Generalized Dimension Exchange Method for Dynamic Load Balancing[*]

C.Z. Xu[†]and F.C.M. Lau

Department of Computer Science

The University of Hong Kong

Abstract

The dimension exchange method is a distributed load balancing method for point-to-point networks. We add a parameter, called the *exchange parameter*, to the method to control the splitting of load between a pair of directly connected processors, and call this parameterized version the *generalized dimension exchange* (GDE) method. The rationale for the introduction of this parameter is that splitting the workload into equal halves does not necessarily lead to an optimal result (in terms of the convergence rate) for certain structures. We carry out an analysis of this new method, emphasizing on its termination aspects and potential efficiency.

Given a specific structure, one needs to determine a value to use for the exchange parameter that would lead to an optimal result. To this end, we first derive a sufficient and necessary condition for the termination of the method. We then show that equal splitting, proposed originally by others as a heuristic strategy, indeed yields optimal efficiency in hypercube structures. For chains, rings, meshes, and tori, however, optimal choices of the exchange parameter are found to be closely related to the scale of these structures.

Finally, to further investigate the potential of the GDE method, we extend it to allow exchange parameters of different values to be used over the set of edges, and based on this extension, we compare the GDE method with the diffusion method.

[*]Journal of Parallel and Distributed Computing 16, 385–393 (1992)

[†]This research was supported by LI KA SHING Postgraduate Scholarship and the Hong Kong and China Gas Company Limited Postgraduate Scholarship

1 Introduction

In distributed systems, load balancing is an essential technique for spreading out the total workload evenly across the available processors to achieve better performance of distributed computations. Load balancing schemes can be categorized into static and dynamic. In this study, we restrict our attention to dynamic load balancing as it appears to have a greater potential than its static counterpart for producing better results. With dynamic load balancing, workload may migrate from one processor to another during runtime. Detailed discussions and surveys of load balancing in distributed systems can be found in [11, 3].

In the past, numerous dynamic load balancing schemes with different characteristics have been proposed, such as the bidding algorithm [9, 21, 23], the drafting algorithm [18], and the gradient-model algorithm [16, 17]. Most of them were evaluated using simulation in which the performance of the proposed algorithm is dependent upon a number of parameters that characterize the underlying system and computational model. To thoroughly evaluate and compare these algorithms requires a careful selection of a number of combinations of values for these parameters, which could turn out to be a non-trivial task [5, 15, 8]. Therefore, simulation alone is not sufficient for a thorough understanding of a load balancing policy, especially of its more fundamental properties. In fact, the fundamental properties, such as termination, potential efficiency and stability, of a load balancing policy can be studied more effectively using a purely theoretical approach. Some important theoretical works have surfaced in recent years. Cybenko, and later on Bertsekas and Tsitsiklis, analyzed two methods for dynamic load balancing, the *diffusion method* and the *dimension exchange method*, using matrix iterative approach, and derived a sufficient and necessary condition for their termination [6, 2]. Kimura and Ichiyoshi investigated the optimal efficiency of a multi-level schemes for OR-parallel programs using probablity theory [14]. Casavant developed a tool, called *communicating finite automata*, which is a combination of finite automata and directed graphs, to formally model the general behavior of distributed scheduling [4]. Stankovic addressed the issue of stability of distributed scheduling [22]. Among them, Cybenko's matrix iterative approach stands out as being most suited for analyzing the efficiency and termination properties of a load balancing policy, which are the properties we are most interested in. In this study, we concentrate on the dimension exchange method and its analysis in a fashion similar to that of [6, 13].

In the dimension exchange method [20, 6, 13], load balancing happens in one dimension at a time, where a dimension corresponds to some subset of all pairs of directly connected processors, and the result is an *equal* distribution of workload between every pair of processors in this subset. Every dimension would take its turn, and the whole process repeats until some satisfactory balanced state is reached. Application of this method in hypercubes was studied empirically by Ranka *et al.* [20] and analyzed by Cybenko [6]. Later on, Hosseini *et al.* extended it for arbitrary structures using the technique of edge-coloring of graphs [13]. Unfortunately, none of these works addressed the fact that equal splitting of the workload between two directly connected processors at each step does not necessarily lead to an optimal efficiency (the time needed to converge to a balanced state) for certain structures. For some structures, *non-equal* splitting of workload between a pair of processors would yield better results. Section 4 of this paper shows that this is the case for the chain, the ring, the mesh, and the torus. In the sequel, we refer to the splitting of workload between a pair of processors as the *exchange pattern*.

In view of the possibility that non-equal splitting of workload might lead to better efficiency in certain structures, we examine in detail in this paper the relationship between the efficiency of the exchange method and the exchange pattern. We introduce an *exchange parameter* to characterize the exchange pattern. As the value of this parameter can be quite varied (instead of always $1/2$ for equal splitting in past cases), we call our dimension exchange method the *generalized dimension exchange* (GDE) method. We analyze the termination condition and the effect of the exchange parameter on the potential efficiency of GDE. The analysis would not only help us understand more deeply the dimension exchange policy itself, but also aid in the design of more efficient policies for various structures.

The model of the underlying system and computation in this study is similar to that in [6, 13]. Specifically, the system we consider is composed of a finite set of autonomous, homogeneous processors connected by a point-to-point communication network. Each processor has a number of bidirectional communication links through which the processor interacts synchronously with its neighbors. The system can be depicted as a simple connected graph $G = (V, E)$, where V is a set of vertices labeled from 0 to $n-1$ and $E \subseteq V \times V$ is a set of edges. Each vertex represents a processor and each edge $(i, j) \in E$ represents a communication link between processor i and j. The underlying computation is assumed to comprise a large number of independent processes. The basic unit of workload is a process, and one or more processes may be running in a processor at any time. The total workload in an instance of load balancing is assumed fixed—*i.e.*, no processes are created or killed during the period. This model of fixed total workload is the same as the one used in [13], which should be a reasonably close representative of the dynamic situation.

The rest of the paper is organized as follows. The GDE method and its iterative model are presented in Section 2. Section 3 presents the convergence analysis of the iterative process with emphasis on the effects of the exchange parameter. In Section 4, we extend the GDE method with multiple parameters instead of a single parameter for the exchange patterns over the network, and establish the relationship between the GDE method and the diffusion method. We conclude in Section 5 with some remarks on the application of the GDE method.

2 The Generalized Dimension Exchange Method

Similar to the original dimension exchange method for arbitrary structures [13], the GDE method operates on color graphs derived from edge-coloring of the given *system graphs*—the simple connected graphs of the given structures. Consider a system graph $G = (V, E)$. By edge-coloring, the edges of G are colored with some minimum number of colors (say, k) 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. A k-color graph is therefore k-dimensional, and load balancing in the GDE method happens in one dimension at a time. Let $\delta(i)$ denote the degree of a vertex i in G and $\delta(G)$ denote the maximum of the degrees of G's vertices. It is known that the minimum number of colors k is strictly bounded by $\delta(G)$, and $\delta(G) \leq k \leq \delta(G) + 1$ [10].

We index the colors in a given k-color graph with integers from 1 to k. Consequently, the k-color graph can be represented as $G_k = (V, E_k)$, of which E_k is a set of 3-tuples of the form $(i, j; c)$, $(i, j; c) \in E_k$ if and only if c is the color number of the edge $(i, j) \in E$. Figure 1 shows two examples of color graphs for a ring structure of scale 3 and a chain of scale 4 respectively. The integers in parentheses are the assigned color numbers.

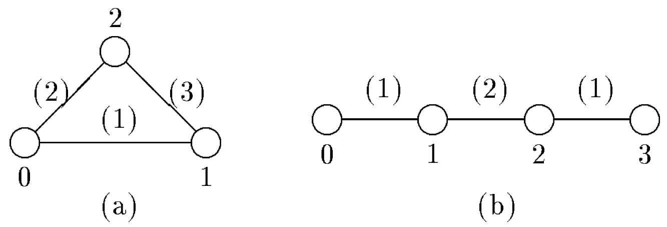

Figure 1: Examples of colored graphs

For a given G_k, let w denote the current local workload of a processor and λ denote the exchange parameter chosen. Then, the change of w in the processor using the GDE method is governed by the algorithm as shown in Figure 2. By executing the operator *Exchange(c)*, a processor sends its local workload to and receives workload from its neighbor along the channel colored c. The variable *InputLoad* temporarily stores the neighboring workload received in the current exchange operation.

```
Procedure for Processor $i$ $(0 \le i < n)$
    while (not Terminate) {
        for $(c = 1; c \le k; c++)$ {
            if there is an incident edge colored $c$ then {
                InputLoad = Exchange(c);
                $w = (1 - \lambda) \times w + \lambda \times InputLoad$;
            }
        }
    }
```

Figure 2: The algorithm of the GDE method

This algorithm is to be run in each processor in a fully decentralized manner, and a processor finishes a complete sweep (*i.e.*, one iteration of the while-loop) after k consecutive exchange operations. As such, the processor interacts with all of its neighbors one at a time in each sweep. In order to guarantee $w \ge 0$, the domain of the exchange parameter λ is restricted to [0,1]. By choosing $\lambda = 1/2$, the GDE method is equivalent to the original dimension exchange method.

Two issues need to be addressed. One is the *termination condition* under which the algorithm would terminate after some finite period of time. The other is the issue of *efficiency*—that is, how many iterative steps are needed for the algorithm to arrive at its termination state. In the following, we provide the answers through analyzing this algorithm using the matrix iterative approach. In discussing efficiency, however, instead of working out the actual number of steps for a given structure, we are interested in the optimal value for the exchange parameter that would lead to the fastest convergence.

With the matrix iterative approach, the workload distribution at some time instant is represented by a vector of variables each of which corresponds to the local workload maintained by a processor; and the change to be applied to the distribution according to the balancing policy is represented by a transformation matrix. The whole process of load balancing is then modeled as an iterative process based on the matrix, which is the center of our analysis.

Let w_i^t $(1 \leq i < n)$ denote the local workload of processor i at time t. Then, the workload distribution at time t is denoted by the vector $W^t = (w_0^t, w_1^t, \ldots, w_{n-1}^t)^T$, and W^0 is the initial workload distribution. Based on the above algorithm, the change of workload in processor i at step t with $c = (t \bmod k) + 1$ can be modeled as

$$\begin{array}{ll} w_i^{t+1} = (1-\lambda) w_i^t + \lambda w_j^t & if\ \exists_j\ (i,j;c) \in E_k \\ w_i^{t+1} = w_i^t & otherwise \end{array} \tag{1}$$

where $0 \leq \lambda \leq 1$.

As a whole, the change of the workload distribution of the entire system at time t with $c = (t \bmod k) + 1$ can be modeled as

$$W^{t+1} = M_c(\lambda) W^t \quad 1 \leq c \leq k \tag{2}$$

where, for $0 \leq i, j < n$ and $i \neq j$,

$$\begin{array}{ll} (M_c(\lambda))_{ii} = (M_c(\lambda))_{jj} = (1-\lambda),\ (M_c(\lambda))_{ij} = (M_c(\lambda))_{ji} = \lambda & if\ (i,j;c)\ \in G_k \\ (M_c(\lambda))_{ii} = 1,\ (M_c(\lambda))_{ij} = 0 & otherwise \end{array}$$

Consequently, the change of the workload distribution of the entire system at time t can be modeled as

$$W^{t+k} = M(\lambda) W^t \tag{3}$$

which implies

$$W^{tk} = M^t(\lambda) W^0 \quad t = 0, 1, 2, \ldots \tag{4}$$

where $M(\lambda) = M_k(\lambda) \times M_{k-1}(\lambda) \times \ldots \times M_1(\lambda)$ is called the *GDE matrix* of the k-color graph G_k. Clearly, each element of $M(\lambda)$ is a polynomial in λ, and by choosing λ=1/2, $M(\lambda)$ reduces to the dimension exchange matrix as shown in [13].

With the above formulation, the features of the GDE method are fully captured by the iterative process governed by $M(\lambda)$. Correspondingly, the termination issue of the algorithm is reduced to the convergence of the sequence $\{M^t(\lambda)\}$, and the efficiency of the algorithm is reflected by the asymptotic convergence rate, $R_\infty(M(\lambda))$.

3 Analysis of the GDE Method

The analysis of the sequence $\{M^t(\lambda)\}$ is divided into two major components. One concerns the condition for termination with respect to the exchange parameter. The other concerns the effects of λ on the convergence rate.

3.1 Convergence

We first consider some important properties of M(λ), which are essential in the ensuing analysis.

Lemma 3.1 *Let G_k be a k-color graph, and $M(\lambda)$ be the GDE matrix of G_k.*

1. *If $0 \leq \lambda \leq 1$, then $M(\lambda)$ is nonnegative and doubly stochastic—that is, for all $1 \leq i, j < n$, $m_{ij}(\lambda) \geq 0$, and $\sum_{1 \leq j < n} m_{ij}(\lambda) = \sum_{1 \leq i < n} m_{ij}(\lambda) = 1$.*
2. *If $0 < \lambda < 1$, then $M(\lambda)$ is primitive—that is, there exists a positive integer s such that $M^s(\lambda) > 0$.*

Proof. (1) Suppose $0 \leq \lambda \leq 1$. By definition, $M_c(\lambda)$ is nonnegative and doubly stochastic for all c, $1 \leq c \leq k$. It is easy to show that their product (of multiplication) preserves the same properties; that is, $M(\lambda)$ is nonnegative and doubly stochastic [1]. (2) Suppose $0 < \lambda < 1$. Then $m_{ii}(\lambda) > 0$ because $(M_c(\lambda))_{ii} > 0$ for all $1 \leq c \leq k$. Hence, $M^{n-1}(\lambda) > 0$; and $M(\lambda)$ is primitive for $0 < \lambda < 1$. □

Because of the nonnegative and doubly stochastic properties, we can analyze the convergence of $\{M^t(\lambda)\}$ using the theory of nonnegative matrices and finite Markov chain [1]. Let $\mu_j(M(\lambda))$ $(1 \leq j \leq n)$ be the eigenvalues of M(λ), and $\rho(M(\lambda))$ be the spectral radius of M(λ), *i.e.*, $\rho(M(\lambda)) = \max_{1 \leq j \leq n}\{|\mu_j(M(\lambda))|\}$. We define

$$\gamma(M(\lambda)) = \max_{1 \leq j \leq n}\{|\mu_j(M(\lambda))| : \mu_j(M(\lambda)) \neq 1\}$$

Then, if $\rho(M(\lambda)) < 1$, $\gamma(M(\lambda)) = \rho(M(\lambda))$; otherwise, $\gamma(M(\lambda))$ is the subdominant eigenvalue of M(λ) in modulus. According to the matrix iterative theory [1], the sequence $\{M^t(\lambda)\}$ is convergent if and only if $\gamma(\lambda) < 1$. Along this line, we establish the following convergence theorem.

Theorem 3.1 *Let $M(\lambda)$ be the GDE matrix of a k-color graph G_k, $\overline{M}$ be a matrix of order n with all elements equal to 1/n, and $\overline{W}$ be a uniform distribution vector with all elements equal to 1. Then,*

1. *$\lim_{t\to\infty} M^t(\lambda) = \overline{M}$ if and only if $\lambda \in (0, 1)$.*
2. *For any initial workload distribution W^0, the sequence $\{W^{tk}\}$ generated by our GDE method converges to $b\overline{W}$ if $\lambda \in$ (0,1), where $b = \sum_{0 \leq i < n}(w_i^0)/n$.*

Proof (1) First, suppose $\lambda \in$ (0,1). Then, from Lemma 3.1, $M(\lambda)$ is nonnegative and primitive. According to the fundamental Perron-Frobenius theorem on nonnegative matrices, we have $\rho(M(\lambda)) > 0$ and its algebraic multiplicity is equal to 1. Also, since $M(\lambda)$ is doubly stochastic, $\rho(M(\lambda)) = 1$. Therefore, $\gamma(M(\lambda)) < 1$. Hence, $\lim_{t\to\infty} M^t(\lambda)$ exists, and each column of the limit matrix is a positive eigenvector of M(λ) corresponding to the spectral radius $\rho(M(\lambda))$. That is, $\lim_{t\to\infty} M^t(\lambda) = \overline{M}$. On the other hand, suppose $\lambda = 0$. Then, $M_c(\lambda)$ is the identity matrix of order n, for all $1 \leq c \leq k$; so is $M(\lambda)$. Suppose $\lambda = 1$. Then $M(\lambda)$ is a permutation matrix because for all $1 \leq c \leq k$, $M_c(\lambda)$ is a permutation matrix. Therefore, $M(\lambda)$ has the eigenvalue 1 in modulus with multiplicity n when $\lambda = 0$ or 1, and hence $\gamma(M(\lambda)) = 1$. Thus, $\lim_{t\to\infty} M^t$ does not exist or if it exists, does not converge to $\overline{M}$. Consequently, $\lim_{t\to\infty} M^t = \overline{M}$ if and only if $\lambda \in (0,1)$. (2) Since $\lim_{t\to\infty} M^t(\lambda) = \overline{M}$, for any initial workload distribution W^0, we have $\lim_{t\to\infty} W^{kt} = \overline{M}W^0 = b\overline{W}$, where b = $\sum_{0\leq i<n} w_i^0/$n. □

The above theorem tells us that $0 < \lambda < 1$ is a sufficient and necessary condition for the termination of dynamic load balancing using the GDE method. If $\lambda = 0$, the workload distribution would not change at all after an iteration step, and if $\lambda = 1$, an iteration step would cause a complete swap of the workloads of every pair of vertices and leave the variance of the workload distribution unchanged.

3.2 Convergence rate

The introduction of the exchange parameter λ into the dimension exchange method is not for forcing a convergence which is present in the original method but rather improving the convergence rate. Consider the asymptotic convergence rate $R_\infty(M(\lambda))$. Since the spectral radius of $M(\lambda)$ is unique, the size of the subdominant eigenvalue in modulus $\gamma(M(\lambda))$ directly affects $R_\infty(M(\lambda))$ which is equal to $-\ln\gamma(M(\lambda))$. If $\gamma(M(\lambda)) = 1$, then $R_\infty(M(\lambda)) = 0$, which means that the iterative process is divergent. If $\gamma(M(\lambda)) = 0$, then $R_\infty(M(\lambda))$ is maximum. The problem 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.

$R_\infty(M(\lambda))$ depends not only on the structure of $M(\lambda)$, *i.e.*, the arrangement of positive and zero elements in $M(\lambda)$, but also on the size of each positive element. To examine the effects of λ on $R_\infty(M(\lambda))$, we first work out a computation formula for the positive elements in $M(\lambda)$, which is based on a concept of color paths in color graphs.

Definition 3.1 *Let G_k be a k-color graph of G. A sequence of edges in G_k of the form*

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

is called a **color path** *of length l from i to j if all intermediate vertices $i_s (1 \leq s \leq l-1)$ are distinct and $k \geq c_1 > c_2 > \ldots > c_l \geq 1$. A color path from i to j is said to be* **closed** *if $i = j$. Two color paths from i to j in G_k 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 $\mathcal{P}_{ij}$.*

For example, in the color graph in Figure 1(a), there exist from vertex 2 to vertex 1 two distinct color paths, (2,1;3) and (2,3;3),(3,1;2), of length 1 and 2 respectively. The sequence of edges (1,2;3), (2,0;2), (0,1;1) is a closed color path of length 3 incident on vertex 1. It is clear from the definition that the length of any color path in a k-color graph cannot be larger than k.

The following lemma presents the computation formula for the elements of $M(\lambda)$. Its proof is based on the concept of color paths and the particular features of each $M_c(\lambda)$, which is somewhat tedious. We leave it to the Appendix.

Lemma 3.2 *Let $M(\lambda)$ be the GDE matrix of a k-color graph G_k. If $0 < \lambda < 1$, then for $0 \le i,j < n$*

$$\begin{array}{ll} m_{ij}(\lambda) = \sum_{p \in \mathcal{P}_{ij}}((1-\lambda)^{r_p}\lambda^{l_p}) & i \neq j \\ m_{ii}(\lambda) = (1-\lambda)^{\delta(i)} + \sum_{p \in \mathcal{P}_{ii}}((1-\lambda)^{r_p}\lambda^{l_p}) & \end{array}$$

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

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

and $r_p = \sum_{s=0}^{l_p}(n_s)$, where n_0 is the number of incident edges of i whose color number is larger than c_1; n_{l_p} is the number of incident edge 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 color number is larger than c_{s+1} and smaller than c_s.

Proof. See Appendix □

As an example, let us examine the GDE matrix of the color graph in Figure 1(a).

$$M(\lambda) = \begin{pmatrix} (1-\lambda)^2 & \lambda(1-\lambda) & \lambda \\ \lambda(1-\lambda) + \lambda^2(1-\lambda) & \lambda^3 + (1-\lambda)^2 & \lambda(1-\lambda) \\ \lambda(1-\lambda)^2 + \lambda^2 & \lambda(1-\lambda) + \lambda^2(1-\lambda) & (1-\lambda)^2 \end{pmatrix}$$

Since there are no closed paths incident on vertices 0 and 1, $m_{00}(\lambda) = m_{22}(\lambda) = (1-\lambda)^2$. However, $m_{11}(\lambda) = \lambda^3 + (1-\lambda)^2$ because of the closed color path of length 3 incident on vertex 1. Consider the ordered pair of vertices $< 2, 0 >$, there are two color paths (2,0;2) and (2,1;3),(3,0;1) from 2 to 0, which contribute to the first and second terms in $m_{20}(\lambda)$, respectively.

We next prove, for a given k-color graph, the existence of an optimal λ that maximizes the asymptotic convergence rate $R_\infty(M(\lambda))$.

Lemma 3.3 *Let $M(\lambda)$ be the GDE matrix of G_k. The eigenvalues of $M(\lambda)$ are a continuous function of λ, $\lambda \in$ [0,1].*

Proof. Because the eigenvalues of $M(\lambda)$ are just the zeros of its characteristic polynomial, they are continuously dependent on the coefficients of the polynomial according to the fundamental theorem of algebra [12]. Given also the fact that the coefficients of the characteristic polynomial of a square real matrix are continuous functions of the elements of the matrix, it follows that the eigenvalues of $M(\lambda)$ are continuously dependent on the elements of $M(\lambda)$. On the other hand, each element of $M(\lambda)$, which is a polynomial in λ according to the last lemma, continuously depends on λ. Therefore, the eigenvalues of $M(\lambda)$ continuously depend on λ. □

Theorem 3.2 *Let $M(\lambda)$ be the GDE matrix of G_k. There exists a $\lambda_b \in$ (0,1) such that $R_\infty(M(\lambda_b)) \ge R_\infty(M(\lambda))$ for all $\lambda \neq \lambda_b$.*

Proof. From Lemma 3.3, $\gamma(M(\lambda))$ is continuously dependent on λ, where $\lambda \in$ [0,1]. Therefore, there exists a $\lambda_b \in$ [0,1] such that $\gamma(M(\lambda_b)) \le \gamma(M(\lambda))$ for all $\lambda \in$ [0,1]. In addition, $\gamma(M(\lambda)) = 1$ if $\lambda = 0$ or 1; otherwise $\gamma(M(\lambda)) < 1$. Hence, $\lambda_b \in$ (0,1), and there exists a $\lambda \in (0,1)$ such that $\gamma(M(\lambda))$ is smallest, or $R_\infty(M(\lambda))$ is largest. □

Our objective is to determine the optimal λ in order to minimize $\gamma(M(\lambda))$. However, as we found no effective methods available for arbitrary matrices, we limit our scope to particular classes of matrices which correspond to particular kinds of structures. Along this line, we establish below a sufficient condition for a k-color graph under which $R_\infty(M(\lambda))$ is largest when $\lambda = 1/2$.

Theorem 3.3 *Let G_k be a k-color graph of G. If G is regular with $\delta(G) = k$, and $|\mathcal{P}_{ij}| = |\mathcal{P}_{ii}| + 1 = |\mathcal{P}_{jj}| + 1$ for each pair of vertices i, j, $0 \le i, j < n$ and $i \ne j$, then $\lambda = 1/2$ is the optimal choice.*

Proof. Suppose $M(\lambda)$ is the GDE matrix of the color graph G_k. If G is regular with $\delta(G) = k$, then the incident edges of each vertex have consecutive color numbers from 1 to k. Hence, the computation formula in the above lemma can be simplified as

$$\begin{array}{ll} m_{ij}(\lambda) = \sum_{p\in\mathcal{P}_{ij}}((1-\lambda)^{k-l_p}\lambda^{l_p}) & i \ne j \\ m_{ii}(\lambda) = (1-\lambda)^{\delta(i)} + \sum_{p\in\mathcal{P}_{ii}}((1-\lambda)^{k-l_p}\lambda^{l_p}) & \end{array}$$

It can be further simplified as $m_{ij}(\lambda) = \sum_{p\in\mathcal{P}_{ij}} \frac{1}{2^k}$ and $m_{ii}(\lambda) = \frac{1}{2^k} + \sum_{p\in\mathcal{P}_{ii}} \frac{1}{2^k}$ for $0 \le i, j < n, i \ne j$ when $\lambda = 1/2$. Furthermore, if $|\mathcal{P}_{ij}| = |\mathcal{P}_{ii}| + 1 = |\mathcal{P}_{jj}| + 1$ $(= s)$, $M(\lambda)$ is then reduced to a uniform matrix, each element of which is equal to $\frac{s}{2^k}$. As a result, the rank of $M(\lambda)$ equals 1, and all the eigenvalues except $\rho(M(\lambda))$ are equal to 0. Thus, $\gamma(M(\lambda)) = 0$, which maximizes the convergence rate. □

With this theorem, we now derive an important result about hypercube structures. Since the hypercube structure is a uniquely colorable graph, *i.e.*, there is only one way of coloring the edges (without respect to the permutation of colors), its corresponding color graph is implicit in the following corollary.

Corollary 3.1 *With the GDE method, $\lambda = 1/2$ is the optimal choice for dynamic load balancing in hypercube structures.*

Therefore, $\lambda = 1/2$, *i.e.*, equal exchange of load between two directly connected nodes, which was initially proposed as a heuristic choice for hypercube structures [20, 6], is now confirmed by this corollary to be the optimal choice. In addition to the hypercube, there are other structures that have the same property. Two examples are the 4×4 torus and the complete graph of 4 vertices.

For the structures of chain, ring, mesh and torus, we examine the spectrums of their GDE matrices, $MC_n(\lambda)$, $MR_n(\lambda)$, $MM_{n_1,n_2}(\lambda)$ and $MT_{n_1,n_2}(\lambda)$ respectively, and calculate their sub-dominant eigenvalues in modulus as λ varies from 0.1 to 0.9 in steps of 0.05. It is found that for a given structure, the optimal exchange parameter λ_b is critically dependent on the scale of the structure. For example, in the chain of 4 vertices, $\lambda_b(MC_4(\lambda))$ is somewhere between 0.55 and 0.65; in the chain of 8 vertices, $\lambda_b(MC_8(\lambda))$ is between 0.7 and 0.8. Furthermore, through closely analyzing these GDE matrices using block circulant matrices [7], we derived the optimal exchange parameters for the "even" case of these structures and uncovered the relationships between their convergence rates. Suppose n_1, n_2 are even, and $n = \max\{n_1, n_2\}$. Then,

$$\lambda_b(MR_{2n}(\lambda)) = \lambda_b(MT_{2n_1,2n_2}(\lambda)) = \lambda_b(MC_n(\lambda)) = \lambda_b(MM_{n_1,n_2}(\lambda)) \qquad (5)$$

which is equal to $\frac{2-\sqrt{2(1-\cos(2\pi/n))}}{1+\cos(2\pi/n)}$. The complete detailed proof can be found in a related paper [24].

4 Extension and Comparison with The Diffusion Scheme

An obvious extension of the method is to relax the restriction of a single parameter and to allow different values of the exchange parameter to be used for different edges. We analyze such an extension in this section and then use it to establish a relationship between the GDE method and the diffusion scheme of Cybenko.

4.1 Extension of The GDE Method

This extension is easily justified by intuition. For instance, consider a chain. One would tend to think that, for better efficiency, the nodes at the two ends should send a major chunk (*i.e.*, a large λ) of its load away during an exchange step, while the nodes at the middle should use a more moderate λ.

We introduce a set of parameters, λ_{ij}, $0 \leq i, j < n$ and $\lambda_{ij} = \lambda_{ji}$, to characterize the set of exchange patterns over all pairs of directly connected processors. Consequently, the basic iterative operation between the two nodes of a colored edge (i,j;c) is modified as

$$w_i = (1 - \lambda_{ij})w_i + \lambda_{ij}w_j$$

Denote Λ to be the set of parameters λ_{ij}. Then, by repeating the same matrix modeling technique as in Section 2, we obtain an extended GDE matrix $M(\Lambda)$, each element of which is a polynomial of λ_{ij}. It is clear that $M(\Lambda)$ is nonnegative and doubly stochastic when $0 \leq \lambda_{ij} < 1$, and primitive when $0 < \lambda_{ij} < 1$ for $0 \leq i, j < n$. Similar to Theorem 3.1, we obtain the following convergence theorem.

Theorem 4.1 *Let $M(\Lambda)$ be the extended GDE matrix of a k-color graph G_k, $\overline{M}$ be a matrix of order n whose elements are all $1/n$, and $\overline{W}$ be a uniform distribution vector whose elements are all 1, then*

1. *$lim_{t\rightarrow\infty} M^t(\Lambda) = \overline{M}$ if and only if $\lambda_{ij} \in$ (0,1) for $0 \leq i, j < n$.*
2. *For any initial workload distribution W^0, the sequence $\{W^{tk}\}$ generated by the extended GDE method converges to $b\overline{W}$ if $\lambda_{ij} \in$ (0,1), for all $0 \leq i, j < n$, where $b = \sum_{1 \leq i < n}(w_i^0)/n$.*

Similarly Theorem 3.2 can be generalized such that for a given color graph G$_k$, there exists a vector of λ_{ij} that maximizes the convergence rate of $M(\Lambda)$.

To further illustrate the relationship between the convergence rate of $M(\Lambda)$ and the parameters λ_{ij}, we analyze quantitatively the extended GDE matrix on the color chain of order 4 as in Figure 1(b). With the extended GDE method, three parameters λ_1, λ_2 and λ_3 are necessary for parameterizing the exchange patterns for edges (0,1), (1,2) and (2,3) respectively. Then the extended GDE matrix is as follows.

$$MC_4(\Lambda) = \begin{pmatrix} 1-\lambda_1 & \lambda_1 & 0 & 0 \\ \lambda_1(1-\lambda_2) & (1-\lambda_1)(1-\lambda_2) & \lambda_2(1-\lambda_3) & \lambda_2\lambda_3 \\ \lambda_1\lambda_2 & \lambda_2(1-\lambda_1) & (1-\lambda_2)(1-\lambda_3) & \lambda_3(1-\lambda_2) \\ 0 & 0 & \lambda_3 & 1-\lambda_3 \end{pmatrix}$$

Similar to the quantitative computation imposed on chain structures, we calculate the subdominant eigenvalue in modulus, $\gamma(M(\Lambda))$, as λ_i $(i = 1,2,3)$ varies from 0.1 to 0.9 in steps of 0.05. It is found that $\gamma(M(\Lambda))$ equals 0.1, the smallest value in the testing group, when $(\lambda_1, \lambda_2, \lambda_3) = (0.5,0.7,0.6)$ or $(0.6,0.7,0.5)$. Theoretically, the smallest value of $\gamma(MC_4(\lambda))$ is equal to $2\lambda - 1 = 0.172$ when $\lambda = 2 - \sqrt{2}$, and hence the extended GDE method using these two vectors of λ_i converges faster than the GDE method with a single, optimal λ.

The dependence of $\gamma(M(\lambda))$ on λ_{ij}, being an optimization problem with multiple parameters, is somewhat difficult to analyze. Nevertheless, in the next subsection, we will show that it is equivalent to the analysis of the diffusion method of Cybenko.

4.2 Comparison with The Diffusion Scheme

In the synchronous diffusion scheme [6], each processor is supposed to interact with all its neighbors at every step, which at step t can be modeled as

$$w_i^{t+1} = w_i^t + \Sigma_{0 \le j < n-1} \alpha_{ij}(w_j^t - w_i^t) \quad 0 \le i < n \tag{6}$$

where α_{ij} is called the diffusion parameter and $0 < \alpha_{ij} < 1$ if nodes i and j are direct neighbors; $\alpha_{ij} = 0$, otherwise. As a whole, the change of the workload distribution at step t is modeled as

$$W^{t+1} = DW^t \tag{7}$$

where D is called a diffusion matrix, and $(D)_{ij} = \alpha_{ij}$ if $i \neq j$; $(D)_{ii} = 1 - \Sigma_{0 \le j < n, j \neq i}(\alpha_{ij})$, otherwise.

Notice that $M(\Lambda)$ reflects a complete sweep—*i.e.*, k consecutive iterative steps, each of which involves an I/O communication action at a processor. On the other hand, the diffusion matrix D reflects a single iterative step which involves $\delta(G)$ I/O communication actions at a processor. Since $\delta(G) \le k \le \delta(G) + 1$ [10], a complete sweep of $M(\Lambda)$ is comparable to an iterative step of D in terms of I/O communication actions.

To facilitate our establishing the relationship between the two methods, we introduce the concept of an extended digraph based on the color graph G_k.

Definition 4.1 *An extended digraph, denoted $\widetilde{G}_k = (V_k, \widetilde{E}_k)$, is a directed graph deduced from a k-color graph $G_k = (V_k, E_k)$ such that*

1. $<i, i> \in \widetilde{E}_k$ *for all* $0 \le i < n$
2. $<i, j> \in \widetilde{E}_k$ *if there exists a color path from i to j in* G_k.

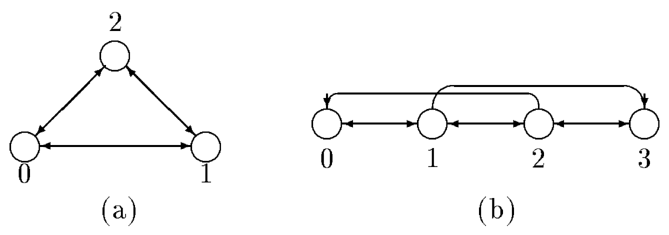

Figure 3: Examples of extended digraphs

Figure 3 depicts two extended digraphs derived from the color graphs in Figure 1.

It is clear from the definition that G is an edge-deleted subgraph of $\widetilde{G}_k$ when each undirected edge $(i, j) \in E$ is viewed as a pair of directed edges $< i, j >$ and $< j, i >$. Now we are ready to show that the GDE matrix $M(\Lambda)$ of a G_k is a diffusion matrix of $\widetilde{G}_k$.

Theorem 4.2 *Suppose $M(\Lambda)$ is the GDE matrix of a k-color graph G_k, and $\widetilde{G}_k$ is the extended digraph of G_k. Then, $M(\Lambda)$ is a diffusion matrix of the digraph $\widetilde{G}_k$.*

Proof. For $0 \le i, j < n$ and $i \neq j$, $m_{ij}(\Lambda) = 0$ if and only if there are no color paths from i to j (from Lemma 3.2), and if and only if $< i, j > \notin \widetilde{G}_k$ (by the definition of $\widetilde{G}_k$). Hence, for $0 \le i, j < n$ and $i \neq j$, $m_{ij}(\Lambda) > 0$ if and only if there is a directed edge $< i, j > \in \widetilde{E}_k$, Moreover, there exists a loop for each vertex, which makes $m_{ii}(\Lambda) > 0$ for $0 \le i < n$. Thus, $M(\Lambda)$ is a diffusion matrix on the digraph $\widetilde{G}_k$. □

The above theorem reveals the similarity between the diffusion and the dimension exchange methods. According to this theorem, the comparison of the two methods is then reduced to the comparison of the diffusion method on two different graphs G_k and $\widetilde{G}_k$. Since G_k is an edge-deleted subgraph of $\widetilde{G}_k$, the diffusion matrix of $\widetilde{G}_k$ has more positive elements than the diffusion matrix of G_k. It follows that a processor balances its workload only with its neighbors at each iterative step in the diffusion scheme while in the dimension exchange method it might balance its workload with more than just its neighbors at each complete sweep. As an example, consider load balancing of the color graph in Figure 1(b). Suppose at the time when the load balancing is activated, processor 0 is most heavily loaded and the others are lightly loaded or empty; processor 0 would therefore diffuse some portion of its workload to its neighbors, and processor 2, for instance, would receive the load in two iterative steps of the diffusion scheme. However, with the GDE method, processor 2 would receive the load in a single complete sweep.

5 Concluding Remarks

We presented in this paper the generalized dimension exchange method (GDE) for dynamic load balancing in distributed systems which is a refinement of the original dimension exchange method with the addition of an exchange parameter. We analyzed its termination property and potential efficiency. The results are: a sufficient and necessary condition for the termination of the GDE scheme, a proof that equal splitting of workload achieves optimal efficiency for the hypercube structure, and in the structures of chain, ring, mesh and torus, that the optimal exchange pattern is critically dependent on their scales. We also extended the GDE method to allow for the use of multiple exchange parameters, and used it to establish the relationship between the GDE model and the diffusion method.

The results derived in this paper not only give us a deeper understanding of the dimension exchange method, but also a basis for designing more efficient methods for specific structures. Like those in [6, 13], our analysis does not take into account the overhead incurred in migrating workload during runtime. However, it is the case in many applications that workloads are represented by only a small amount of information, such as an integer or two, which can be sent around with small or negligible overhead. Note also that in practical implementations, load balancing in our model would be divided into two stages: the first one consists purely of exchanges of load information among the processors until reaching an acceptable balanced state, and then the actual load migration would take place in the second stage.

To complement the GDE method, we have developed a fully distributed mechanism for the termination detection of load balancing based on the method [25]. We are also in the process of incorporating the GDE load balancing algorithm into remapping of data parallel computations [19].

Finally, we should point out that the GDE method is based on the assumption of a synchronous model in which migration of workload across an edge can be treated as instantaneous. In asynchronous models, however, because of the communication delays and asynchronism among the processors, the analysis of applicable load balancing methods is expected to be more difficult than the present analysis. Bertsekas *et al.* ventured into this domain by analyzing the asynchronous diffusion scheme [2].

Acknowledgements. We would like to thank Prof. G. Cybenko for his valuable comments and pointing out the reference [7], and the anonymous referees for their constructive comments.

References

[1] A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences.* Academic Press, 1979.

[2] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods.* Prentice-Hall Inc., 1989.

[3] T.L. Casavant and J.G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Trans. on Software Engineering*, 14:141–154, May 1988.

[4] T.L. Casavant and J.G. Kuhl. A communicating finite automata approach to modeling distributed computation and its application to distributed decision-making. *IEEE Trans. on Computers*, 39(5):628–639, May 1990.

[5] S. Chowdhury. The greedy load sharing algorithm. *J. Parallel and Distributed Computng*, 9:93–99, 1990.

[6] G. Cybenko. Load balancing for distributed memeory multiprocessors. *J. Parallel and Distributed Computing*, 7:279–301, 1989.

[7] P.J. David. *Circulant matrices.* John and Sons Inc., 1979.

[8] D.L. Eager, E.D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. on Software Engineering*, 12, May 1986.

[9] D.J. Faber. The distributed computing systems. *Proc. Compcon Spring*, 31–34, 1973.

[10] S. Fiorini and R.J. Wilson. Edge-coloring of graphs. In L. W. Beineke and R. J. Wilson, editors, *Selected topics in graph theory.* Academic Press, 1978.

[11] A.J. Harget and I.D. Johnson. Load balancing algorithms in loosely-coupled distributed systems: a survey. In H.S.M. Zedan, editor, *Distributed Computer Systems*, 85–108. Butterworths, 1990.

[12] R.A. Horn and C.R. Johnson. *Matrix analysis.* Cambridge University Press, 1985.

[13] S.H. Hosseini, B. Litow, M. Malkawi, J. Mcpherson, and K. Vairavan. Analysis of a graph coloring based distributed load balancing algorithm. *J. Parallel and Distributed Computing*, 10:160–166, 1990.

[14] K. Kimura and N. Ichiyoshi. Probabilistic analysis of the optimal efficiency of the multi-level dynamic load balancing schems. *Proc. 6th Distributed memory computing Conference*, 145–152, Portland, Oregon, USA, April 1991.

[15] T. Kunz. The influence of different workload descriptions on a heuristic load balancing scheme. *IEEE Trans. on Software Engineering*, 17(7):725–730, 1991.

[16] F.C.H. Lin and R.M. Keller. The gradient model load balancing method. *IEEE Trans. on Software Engineering*, 13(1):32–38, Jan. 1987.

[17] R. Luling, B. Monien, and F. Ramme. Load balancing in large networks: A comparative study. In *Proc. 3th IEEE Symposium on parallel and distributed processing*, 686–689, Dallas, Texas, USA, Dec. 2-5, 1991.

[18] L.M. Ni, C-W Xu, and T.B. Gendreau. A distributed drafting algorithm for load balancing. *IEEE Trans. on Software Engineering*, 11(10):1153–1161, Oct. 1985.

[19] D.M. Nicol and P.F. Reynolds. Optimal dynamic remapping of data parallel computation. *IEEE Trans. on Computers*, 39(2):206–219, Feb. 1990.

[20] S. Ranka, Y. Won, and S. Sahni. Programming a hypercube multicomputer. *IEEE Software*, 5:69–77, Sept. 1988.

[21] R. Smith. The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Trans. on Computer*, 27:1104–1113, Dec. 1980.

[22] J.A. Stankovic. Stability and distributed scheduling algorithms. *IEEE Trans. on Software Engineering*, 11(10):1141–1152., Oct. 1985.

[23] J.A. Stankovic and I.S. Sidhu. An adaptive bidding algorithm for processes, clusters and distributed groups. In *Proc. 4th International Conference on Distributed Computer Systems*, 49–59, San Francisco, CA, May 1984.

[24] C.Z. Xu and F.C.M. Lau. The generalized dimension exchange method on some specific structures. Technical Report TR-92-02, Dept. of Computer Science, The University of Hong Kong, Jan. 1992.

[25] C.Z. Xu and F.C.M. Lau. Termination detection for synchronous parallel iterative computations. Technical report, Dept. of Computer Science, The University of Hong Kong, June 1992.

Appendix – The Proof of Lemma 3.2

Given an ordered pair of vertices $< i, j >$ in G_k, we exhaust the three cases concerning the color paths from i to j.

Case 1. There are no color paths from i to j in G_k. Then for all $1 \leq c \leq k$, $(M_c(\lambda))_{ij}$ would be 0, and the element (i, i) in the $\delta(i)$ matrices would equal to $1 - \lambda$, and 1 in the other matrices. From the definition of $M(\lambda)$,

$$M(\lambda) = M_k(\lambda) \times \ldots \times M_c(\lambda) \times \ldots \times M_1(\lambda)$$

It is then clear that $m_{ii}(\lambda) = (1 - \lambda)^{\delta(i)}$; $m_{ij}(\lambda) = 0$, $i \neq j$.

Case 2. There is a single color path of length l from i to j.

1. If $l = 1$, *i.e.*, i and j are adjacent, then $(M_c)_{ij} = (M_c)_{ij} = \lambda$, and $(M_c)_{ii} = (M_c)_{jj} = 1 - \lambda$, where c is the color number of the edge. Because there are no matrices $M_h(1 \leq h \leq k,\ and h \neq c)$ in which the elements in positions $(i, j), (j, i), (i, i)$ and (j, j) have the same size as the corresponding elements in M_c, the following statements can be derived according to the definition of $M(\lambda)$. For all $h, c < h \leq k$, if $(M_h)_{ij} = 1$, *i.e.*, vertex i has no incident edges with color number h, then the premultiplication of $M_c(\lambda)$ by $M_h(\lambda)$ does not have any effects on the size of the element $(M_c(\lambda))_{ij}$; if $(M_h)_{ij} = 1 - \lambda$, *i.e.*, vertex i has an incident edge with color number h, then the premultiplicity of $M_c(\lambda)$ by $M_h(\lambda)$ changes the size of the element $(M_c(\lambda))_{ij}$ to $(1 - \lambda) \times (M_c(\lambda))_{ij}$. Similarly, for all h, $1 \leq h < c$, the postmultiplicity of $M_c(\lambda)$ by $M_h(\lambda)$ changes $(M_c(\lambda))_{ij}$ to $(1 - \lambda)M_c(\lambda)$ only when vertex j has an incident edge with color number h. As a result, we have

$$m_{ij}(\lambda) = (1 - \lambda)^{n_0} \times \lambda \times (1 - \lambda)^{n_1} = (1 - \lambda)^{n_0 + n_1} \times \lambda^l$$

where n_0 is the number of incident edges of i whose color number is larger than c and n_1 is the number of incident edges of j whose color number is smaller than c.

2. If $l > 1$, and the color path is of the form

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

then for all $1 \leq s \leq l$, we have $(M_{c_s}(\lambda))_{ij} = (M_{c_s}(\lambda))_{ji} = \lambda$, and $(M_{c_s}(\lambda))_{ii} = (M_{c_s}(\lambda))_{jj} = 1 - \lambda$. Generally, $M(\lambda)$ is of the form

$$M_k(\lambda) \times \ldots \times M_{c_1}(\lambda) \times \ldots \times M_{c_2}(\lambda) \times \ldots \times M_{c_l}(\lambda) \times \ldots \times M_1(\lambda)$$

From the analysis for the case of $l = 1$, it can be deduced that

$$\begin{aligned} m_{ij}(\lambda) &= (1-\lambda)^{n_0} \times \lambda \times (1-\lambda)^{n_1} \times \lambda \times \ldots \times (1-\lambda)^{n_{l-l}} \times \lambda \times (1-\lambda)^{n_l} \\ &= (1-\lambda)^{n_0+n_1+\ldots+n_l}\lambda^l \end{aligned}$$

where n_0 is the number of incident edges of i whose color number is larger than c_1, n_l is the number of incident edges of j whose color number is smaller than c_l, and $n_s(1 \leq s \leq l-1)$ is the number of incident edges of vertex i_s whose color number is larger than c_{s+1} and smaller than c_s.

Case 3. There are more than one distinct color path from i to j, all together comprising a set $\mathcal{P}_{ij}$. Then according to the principle of matrix multiplication, we have

$$m_{ij}(\lambda) = \textstyle\sum_{p\in\mathcal{P}_{ij}}((1\text{-}\lambda)^{r_p}\lambda^{l_p}) \quad 1 \leq i,j \leq n$$

where l_p and r_p are as stated in the lemma.

Hence, the lemma follows. □