1989

A grid based subtree-subcube assignment strategy for solving PDEs on hypercubes

Mo Mu

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:
89-869

A GRID BASED SUBTREE-SUBCUBE ASSIGNMENT STRATEGY FOR SOLVING PDES ON HYPERCUBES

Mo Mu
J.R. Rice

CSD-TR-869
February 1989

A GRID BASED SUBTREE-SUBCUBE ASSIGNMENT STRATEGY FOR SOLVING PDES ON HYPERCUBES

Mo Mu*
and
J.R. Rice**

Computer Sciences Department
Purdue University
Technical Report CSD-TR-869
CAPO Report CER-89-12
February, 1989

* Work supported in part by National Science Foundation grant CCR-8619817.
** Work supported in part by the Air Force Office of Scientific Research grant, 88-0243, and the Strategic Defense Initiative through Army Research Office contract DAAL03-86-K-0106.

A GRID BASED SUBTREE-SUBCUBE ASSIGNMENT STRATEGY FOR SOLVING PDES ON HYPERCUBES

Mo Mu * and John R. Rice**

Computer Science Department
Purdue University
West Lafayette, IN 47907
Technical Report CSD-TR-869
CAPO Report CER-89-12
February 1989

ABSTRACT

We propose a grid based subtree-subcube assignment strategy for using nested dissection in solving PDE problems on hypercubes. A complexity analysis is given for both our approach and the standard subtree-subcube assignment. The new assignment reduces communication cost by a factor of $O(\log p)$ in start ups and a factor of about two in traffic volume. This grid based assignment strategy achieves the optimal order in both traffic volume and start ups, it provides load balancing and as much parallelism as is inherent in the algorithm formulation.

* Work supported in part by National Science Foundation grant CCR-8619817.
** Work supported in part by the Air Force Office of Scientific Research grant, 88-0243 and the Strategic Defense Initiative Office contract DAAL03-86-K-0106.

I. INTRODUCTION

This paper deals with the assignment issue in solving PDE problems on a hypercube multiprocessor. There have been many parallel algorithms developed for solving sparse matrix problems on hypercubes (e.g., [George, Heath, Liu, Ng, 1988], [Mu, Rice, 1989]). Most of them are based on elimination trees and are intended for linear systems such as those that arise from solving PDEs. An ideal assignment strategy should keep the load balanced, exploit fully the parallelism inherent in the problem and minimize the communication requirement. An attractive **subtree-subcube** approach is proposed by [George, Liu, Ng, 1987] which has the lowest order of traffic volume. However, for most current commercially available hypercube multiprocessors, both the traffic volume and the number of communication start ups affect efficiency very much. We apply an idea from the multifrontal method to minimize the start up cost while also retaining the other desirable properties.

II. BACKGROUND

(a) Elimination trees. A geometric approach to develop parallel solvers for solving PDE problems is given by [Mu and Rice, 1989] which mixes nested dissection with minimum degree ordering using domain decomposition and which tries to exploit their advantages and minimize their disadvantages to some degree. It also allows a lack of symmetry while keeping elimination trees well shaped. For simplicity, we consider a PDE problem on a rectangular domain Ω. The approach can be easily extended to general domains. Suppose we have $p\,(=2^{2d})$ processors available. By domain decomposition Ω is divided into p subdomains Ω_{ij}, $i, j = 1, 2, \ldots, p^{1/2}$ as shown in Figure 2.1.

Ω_{11}	Ω_{12}	Ω_{13}	Ω_{14}
Ω_{21}	Ω_{22}	Ω_{23}	Ω_{24}
Ω_{31}	Ω_{32}	Ω_{33}	Ω_{34}
Ω_{41}	Ω_{42}	Ω_{43}	Ω_{44}

Figure 2.1. Domain decomposition of a rectangle for $d = 2$.

One puts a local grid on each subdomain Ω_{ij} and discretizes the local problem whose solution U_{ij} only depends on unknowns at grid points of $\partial\Omega_{ij}$. First, all interior unknowns U_{ij} are eliminated locally as in the standard domain decomposition approach. This step is obviously totally parallel. Within each subdomain elimination occurs sequentially in a single processor, so in principle any efficient ordering can be used here. Furthermore, if we keep the elimination front in Ω_{ij} as far as possible from $\partial\Omega_{ij}$, it helps to reduce the communication cost. Denote the boundary layer of grid points in Ω_{ij} (those next to $\partial\Omega_{ij}$) by B_{ij}. Without loss of generality assume that only unknowns on B_{ij} are related to those on $\partial\Omega_{ij}$ in the linear system, such as **five point star** would generate. Then we first eliminate the unknowns on Ω_{ij} / B_{ij} using the minimum degree ordering. There is no communication required at this stage. Then unknowns on B_{ij} are eliminated. Second, all processors participate in eliminating interface unknowns. The number of these unknowns is only of order $O(N^{1/2})$ if the total number of unknowns is N. Here parallelism is more important than fill-in and we use one way nested dissection to partition the interface set into several levels suitable for a hypercube machine, each level consists of several groups of nodes in the elimination tree. The partition is shown by Figure 2.2 along with the numbering of the elimination tree nodes. The circles represent the unknowns interior to the subdomain Ω_{ij}, the boxes are the **separators**, groups of unknowns which separate regions in the nested dissection method.

The PDE discretization process leads to a local or boundary dependence property for interfaces. For example, if we consider the union of subdomains 16, 17, 8 as a more general subdomain Ω'_8 then the local interior solution set U'_8 is uniquely determined by unknowns at grid points on $\partial\Omega'_8$. This relation holds similarly for groups at higher levels of the elimination tree for the unknowns arising in PDE applications. Thus we obtain a condensed binary elimination tree as shown in Figure 2.3 which reflects the local dependency. Each node corresponds to a subdomain or a separator in the nested dissection. This elimination tree has the following properties: (a) Each node corresponds to groups of unknowns from one location, local ordering and lack of symmetry do not affect the tree structure, (b) Eliminating a node only has effects on its ascendants, (c) The eliminations of the descendent nodes of a given node are independent of one another.

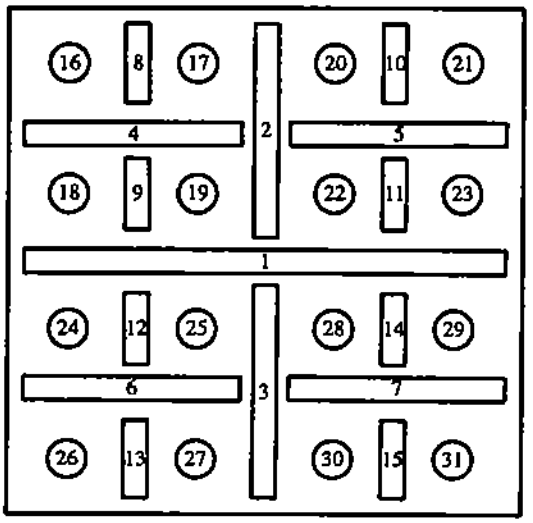

Figure 2.2. Partition of the subdomain interfaces in Figure 2.1 using one way nested dissection. The circles (16–31) represent the 16 groups of interior unknowns and the boxes represent groups of interface unknowns or separators. All boxes of the same size are on the same level of the elimination tree.

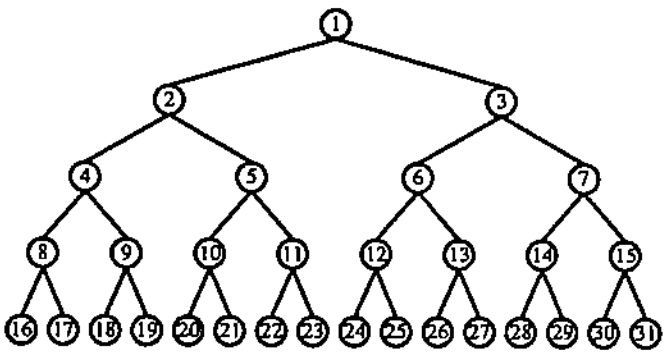

Figure 2.3. Condensed elimination tree produced by domain decomposition and nested dissection. The numbering of nodes corresponds to the groups of unknowns in Figure 2.2.

(b) Potential parallelism. There are four kinds of potential parallelism here. First, elimination steps in independent nodes can execute simultaneously. We call this the **outer parallelism**. Second, if there are several processors available for a single node, we can also exploit **inner parallelism** within the node. This does not occur at

leaf nodes because each leaf node has only one processor even though it represents a sparse subproblem. For the other nodes we apply various efficient parallel dense solvers to exploit the inner parallelism. Third, the modification task in elimination is independent for different equations just as for dense matrices. Finally, fourth, modifications, even on the same equation, due to independent descendent nodes can be performed in arbitrary order.

III. ASSIGNMENT STRATEGY

By assigning a grid point to a processor we mean assigning both the problem data and the factorization subtask associated with this point. To achieve high parallelism, load balancing and cheap communication we want to (a) avoid assigning independent nodes to the same processor, and (b) assign processors to a single node so as to have minimal communication connections. The standard wrapping assignment is not as effective here even though it achieves load balancing.

In [George, Liu, Ng, 1987] an attractive **subtree-subcube with local wrapping** assignment is proposed. We refer to it as the standard subtree-subcube assignment. It is a top to bottom process. First, the root node of the elimination tree is assigned to the whole hypercube and then the hypercube is split into two subcubes to which the two descendent subtrees are assigned. This process goes on recursively until all subtrees become assigned to single processors. The assignment within each node is in wrapping manner. Of course, the above process can be extended to general tree structures. Note that: (a) eliminating an unknown in a node need not affect all of its ancestor nodes, and (b) even when effects occur in some ancestor nodes, they need not affect all equations in them. Geometrically, the effect of elimination spreads in a multifrontal manner. However, one cannot represent these properties completely by elimination trees and yet they may affect the parallel efficiency very much. This suggests that grid point assignment be made in a multifrontal manner with a processor responsible only for those grid points located at the fronts of some nodes to which the processor has been assigned. If several processors (usually a subcube) correspond to the same set of points, then local wrapping can be applied within this set. We call this the **grid based subtree-subcube** assignment which is defined more precisely as follows.

This is a bottom to top assignment process. Let us denote the levels in the elimination tree from bottom to top by 0, 1^{st} x, 1^{st} y, 2^{nd} x, 2^{nd}, ..., i^{th} x, i^{th} y and so on (see Figure 2.2). The first step is to map the given hypercube to a two dimensional grid (for domain decomposition) by the well-known *gray code* such that adjacent processors

are directly connected and Ω_{ij} is assigned to processor P_{ij}. This defines the assignment at the leaves of the elimination tree, the 0 level. Next, we subdivide each separator on the i^{th} x level and the i^{th} y level into 2^{i-1} and 2^i **segments**, respectively. This subdivision of segments corresponds to the natural geometric segments along the separators of the domain decomposition. We take care of the intersection points of adjacent segments in each separator by adding an intersection point to its left (top) segment for the $x(y)$ direction. Then we assign each segment on i^{th} $x(y)$ level to the closest 2^i processors in the $x(y)$ direction. The assignment within each segment uses wrapping. This scheme is illustrated in Figure 3.1. Thus, processors P_{11} and P_{12} are assigned to the interface unknowns of separator 8 (the upper left box in Figure 3.1).

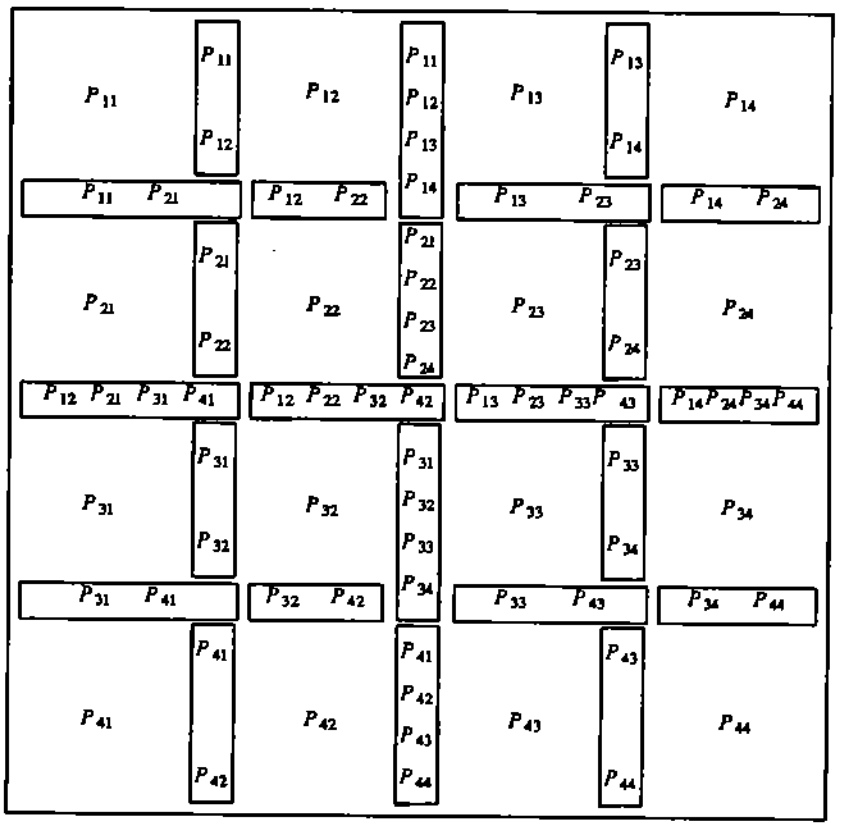

Figure 3.1. Grid based subtree-subcube assignment for 16 processors. Within the subdomain interfaces we show how the processors are assigned to parts of the separators of grid points.

For comparison Figure 3.2 illustrates the standard subtree-subcube assignment strategy.

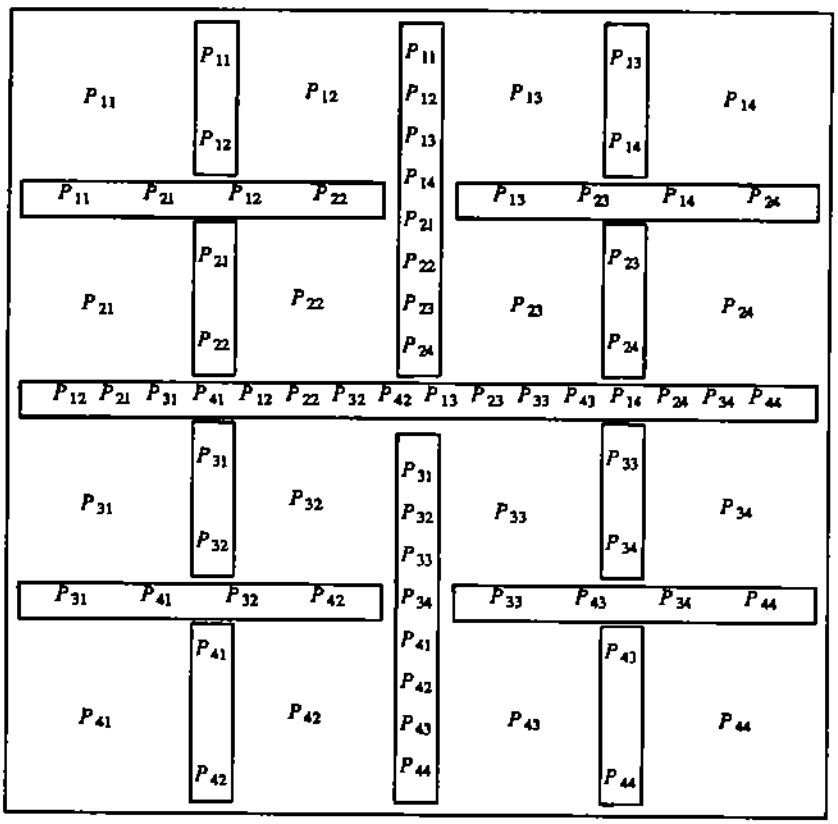

Figure 3.2. Standard subtree-subcube assignment for 16 processors. Within each box grid points are assigned in wrapping manner to processors shown in the box.

The potential for reducing communication is seen by examining separator 4 which is the top, left horizontal box in Figures 2.2 and 3.2. In Figure 3.1 we see those unknowns divided into two separate groups (segments). In the grid based subtree-subcube assignment (Figure 3.1), the processors P_{11} and P_{12} only handle the interface between the two subdomains (16 and 18) they handled at the lower level. In the standard subtree-subcube assignment (Figure 3.2), the processors P_{11} and P_{21} are part of a group of processors handling the four subdomains (16, 17, 18 and 19). Thus P_{11} and P_{21} must now obtain information about subdomains 17 and 19 at this step while this is

not required in the grid based subtree-subcube assignment. This reduction in the communication occurs in a similar manner for every separator.

IV. COMMUNICATION COMPLEXITY ANALYSIS

In [George, Liu, Ng, 1987] it is proved that the total amount (or volume) of communication for the standard subtree-subcube assignment is $O(pN)$. This order is optimal in the sense of minimizing traffic volume for nested dissection algorithms, our grid based subtree-subcube assignment has the same optimal order as George's analysis applies for all types of subtree-subcube assignments. We give an analysis which provides estimates for the communication complexity of start ups as well as for traffic volume. We show that our assignment gains an additional $O(\log_2 p)$ reduction in start up cost compared to the standard assignment. As a by product, we see that our assignment also reduces the volume of communication by a factor of about two.

Suppose that a set of q processors compose a connected subgraph in the connection architecture and they are involved in communicating a piece of message. We assume that each non-root processor receives the message exactly once from one of its direct neighbors. This occurs in most multicasting procedures. Therefore the total number of start ups required in such a communication process is equal to $q - 1$. Putting a $k \times k$ grid on each subdomain Ω_{ij}, we have the total number of grid points on Ω equal to $N = n \times n$ with $n = p^{1/2} k + p^{1/2} - 1$.

Let C_0, C_{1x}, C_{1y}, C_{2x}, C_{2y}, $\cdots$ denote the number of start ups required for communication involved in eliminating unknowns on the 0, $1^{st}x$, $1^{st}y$, $2^{nd}x$, $2^{nd}y$, ..., level, respectively. Actually, there are two types of communication involved in eliminating each unknown. One is for multipliers and the other is for communication information [Mu and Rice, 1989]. We only consider the analysis for the former here as the latter is similar. That is, in eliminating a particular unknown we need to count the cost for communicating a piece of multiplier message (a vector) among those processors holding unknowns which are currently related to this unknown. First we count the start up cost for the grid based subtree-subcube assignment strategy at the bottom level.

Lemma 1: *Let n_o be the number of grid points next to the boundary of a subdomain. Then the number C_o of communication start ups at the bottom level is bound as follows,*

$$C_0 \le (2p\log_2 p - \frac{9}{2}p + 5p^{1/2} + 2)n_o \qquad (4.1)$$

Proof: We bound the cost by the technique of considering each interface separator (numbers 1, 2, ..., 15 seen in Figure 2.2) and count how many start ups are associated with it. Let us first look at the $\ell^{\text{th}}y$ level. There are $4^{d-\ell}$ separators on this level and 2^ℓ segments plus $(2^\ell - 1)$ intersection points in each of these separators. Each segment is assigned to 2^ℓ processors and has subdomains on both sides. Elimination within these subdomains causes communication among processors associated with this segment and the associated intersection point. For the $1^{\text{st}}y$ level, only those separators on the top most and bottom most lines need to be considered because each of the other separators is bordered by two separators on higher levels and it is assigned to a subset of processors for those two separators. The number of these active separators is $2 \cdot 2^{d-1}$. For the same reason we only need to count the number of start ups due to eliminating unknowns on one side. Furthermore, each of these active separators consists of two segments. Elimination effects in level 0 on such a segment involves only one subdomain except for the intersection points which involve two subdomains. Recall that each boundary layer of grid points B_{ij} has $n_o = 4(k-1)$ points. Therefore the cost for eliminating unknowns next to the $1^{\text{st}}y$ level separators, denoted by $C_0^{1,y}$, satisfies

$$C_0^{1,y} \le 2 \cdot 2^{d-1} \Big[2(2-1) + (2-1) \Big] n_o \tag{4.2a}$$

Similarly for unknowns next to the $\ell^{\text{th}}y$ level separators with $\ell \ge 2$, each segment is basicly affected by eliminating two subdomains. So it follows that the cost $C_0^{\ell,y}$ for the $\ell^{\text{th}}y$ level satisfies

$$C_0^{\ell,y} \le 4^{d-\ell} \cdot 2 \Big[2^\ell(2^\ell - 1) + (2^\ell - 1) \Big] n_o, \quad \ell \ge 2 \tag{4.2b}$$

We have similar notation and estimates for the x direction, namely,

$$C_0^{1,x} \le 2 \cdot 2^d n_o \tag{4.2c}$$

$$C_0^{\ell,x} \le 2 \cdot 4^{d-\ell} \cdot 2 \Big[2^{\ell-1}(2^\ell - 1) + (2^{\ell-1} - 1) \Big] n_o, \quad \ell \ge 2 \tag{4.2d}$$

The total number of start ups at the bottom is

$$C_0 = C_0^{1,y} + \sum_{\ell=2}^{d} C_0^{\ell,y} + C_0^{1,x} + \sum_{\ell=2}^{d} C_0^{\ell,x} \tag{4.3}$$

which leads to (4.1) by using the (4.2) estimates. This concludes the proof.

Now we consider the start ups due to eliminating unknowns on the $i^{\text{th}}x$ level.

Lemma 2: *For $i \le d-2$,*

$$C_{ix} \le [\, 4p(d-i-1) + 2p + p^{1/2}(6 \cdot 2^i - 4 \cdot 2^{-i}) - p4^{-i} + 2 \,] n_{ix} \tag{4.4a}$$

and for $i = d-1,\ d,$

$$C_{(d-1)x} \le (5p-10)n_{(d-1)x} \tag{4.4b}$$

$$C_{dx} \le 2(p-1)n_{dx} \tag{4.4c}$$

where $n_{ix} = 2^{i-1}k + 2^{i-1} - 1$ *is the number of grid points in each* $i^{\text{th}}x$ *level separator.*

Proof: Let $C_{ix}^{\ell,x}$ and $C_{ix}^{\ell,y}$ denote the number of start ups for eliminating unknowns next to the $\ell^{\text{th}}x$ and y level separators during the elimination of unknowns on the $i^{\text{th}}x$ level. Then we claim

$$C_{ix}^{i+1,x} \le 2\cdot 2^{d-i}\left[\,2(4^i-1)+(2-1)\,\right]n_{ix} \tag{4.5a}$$

$$C_{ix}^{\ell,x} \le 2\cdot 4^{d-\ell}\cdot 2\left[\,2^{\ell-i}(2^{\ell+i-1}-1)+(2^{\ell-1}-1)\,\right]n_{ix},\quad \ell \ge i+2 \tag{4.5b}$$

$$C_{ix}^{i,y} \le 2\cdot 2^{d-i}(4^i-1)n_{ix} \tag{4.5c}$$

$$C_{ix}^{\ell,y} \le 4^{d-\ell}\cdot 2\left[\,2^{\ell-i}\,(2^{\ell+i}-1)+(2^{\ell-i}-1)\,\right]n_{ix},\quad \ell \ge i+1\,. \tag{4.5d}$$

To see this, we treat separators on the $i^{\text{th}}x$ level as subdomains and treat a group of 2^{i-1} segments (2^i segments) on higher $x(y)$ levels as a more general segment. Each separator on the ℓ^{th} x or y level has $2^{\ell-i}$ such groups and each of these groups on the $\ell^{\text{th}}x(y)$ level has $2^{\ell+i-1}$ ($2^{\ell+i}$) processors. Then an argument similar to that in Lemma 1 can be applied to obtain (4.5).

Notice that for $i \le d-2$,

$$C_{ix} = C_{ix}^{i+1,x} + \sum_{\ell=i+2}^{d} C_{ix}^{\ell,x} + C_{ix}^{i,y} + \sum_{\ell=i+1}^{d} C_{ix}^{\ell,y} \tag{4.6a}$$

and, for $i = d-1$,

$$C_{(d-1)x} = C_{(d-1)x}^{d,x} + C_{(d-1)x}^{(d-1),y} + C_{(d-1)x}^{d,y} \tag{4.6b}$$

Finally, for $i = d$ we have

$$C_{dx} = C_{dx}^{d,y}\quad . \tag{4.6c}$$

Combining (4.5) and (4.6) we get (4.4) and this completes the proof.

Similarly for eliminating unknowns on the $i^{\text{th}}y$ level we have the following.

Lemma 3: *For* $i \le d-2$,

$$C_{iy} \le [4p(d-i-1) + p^{1/2}(8\cdot 2^i - 3\cdot 2^{-i}) - \frac{1}{2}p4^{-i} + 2]n_{iy} \tag{4.7a}$$

and for $i = d-1,\ d,$

$$C_{(d-1)y} \le (4p-6)n_{(d-1)y} \tag{4.7b}$$

$$C_{dy} \le (p-1)n_{dy} \quad . \tag{4.7c}$$

where $n_{iy} = 2^i k + 2^i - 1$ is the number of grid points in each separator on $i^{\text{th}}y$ level

Proof: Let $C_{iy}^{\ell,x}$ and $C_{iy}^{\ell,y}$ denote the number of start ups on the ℓ^{th} x and y levels during the elimination of unknowns in the i^{th} y level. Then we have, for $i < d$,

$$C_{iy}^{i+1,x} \le 2 \cdot 2^{d-i}\,(p_{(i+1)x}^{i} - 1)n_{iy}\ ; \tag{4.8a}$$

$$C_{iy}^{\ell,x} \le 2 \cdot 4^{d-\ell} \cdot 2(2^{\ell-i-1}(p_{\ell x}^{i} - 1) + 2^{\ell-i-1} - 1)n_{iy}, \quad \ell \ge i+2\ ; \tag{4.8b}$$

$$C_{iy}^{i+1,y} \le 2 \cdot 2^{d-i-1}(2(p_{(i+1)y}^{i} - 1) + 1)n_{iy}\ ; \tag{4.8c}$$

$$C_{iy}^{\ell,y} \le 4^{d-\ell} \cdot 2(2^{\ell-i}(p_{\ell y}^{i} - 1) + 2^{\ell-i} - 1)n_{iy}, \quad \ell \ge i+2 \tag{4.8d}$$

where $p_{\ell x}^{i} = p_{\ell y}^{i} = 2^{\ell+i}$ is the number of processors for each segment group (or general segment) in the ℓ^{th} level separators with groups divided related to $i^{\text{th}}y$ level.

To see this we observe that in this case each separator, for $\ell > i$, on the ℓ^{th} $x(y)$ level is divided into $2^{\ell-i-1}(2^{\ell-i})$ groups and each group consists of 2^i segments and therefore has $2^{\ell+i}$ processors. So we get (4.8) by arguments similar to those of Lemma 1.

It is easy to see that

$$C_{iy} = \sum_{\ell=i+1}^{d} (C_{iy}^{\ell,x} + C_{iy}^{\ell,y}) \quad . \tag{4.9}$$

Combining (4.8) and (4.9) we get (4.7a) and (4.7b). For (4.7c) we notice that there are n grid points in the node and p processors participate in communication for eliminating each point. This concludes the proof.

From Lemmas 1 through 3 we conclude:

Theorem 1. *Suppose $d \ge 3$ and there are p processors and N unknowns. The number S_g of start ups required in the grid based subtree-subcube assignment satisfies*

$$S_g \le (\frac{38}{3}p + 2p^{1/2}\log_2 p - 21p^{1/2} - \frac{5}{2}\log_2 p + \frac{71}{2} + 2p^{1/2})N^{1/2} + O(1) \tag{4.10}$$

Proof: We have the following expression for S_g in terms of quantities estimated in the previous lemmas:

$$S_g = C_o + \sum_{i=1}^{d} (C_{ix} + C_{iy})$$

Substituting the results of Lemmas 1 to 3, equations (4.1), (4.4) and (4.7) in the formula for S_g and simplifying completes the proof.

Even though the above analysis gives an upper bound, we can see that the bound is sharp in terms of the orders for N and p for a pipelined algorithm. The final step (node 1 of the elimination tree) involves the $\sqrt{N}$ unknowns on the segment 1, see Figure 2.2. This step involves a dense matrix where all processors must exchange information with all other processors about all the remaining $\sqrt{N}$ equations. If we maintain pipelining, then each multiplier vector must be broadcast as soon as it is available. Thus $\sqrt{N}$ message start ups are required from each processor. One could consider a partially pipelined algorithm where several multiplier vectors are collected before being broadcast. This would be similar in philosophy to loop unrolling (see [Geist and Romine, 1988]), it has the disadvantage of delaying computations on other processors. While the optimal strategy will depend on the exact characteristics of the hypercube, we believe that the optimal strategy will usually involve only a small number of vectors to be collected together before boradcasting so the order will remain the same. Next we consider the standard subtree-subcube assignment strategy.

Theorem 2. *Suppose $d \geq 3$. The number S_s of start ups required in the standard subtree-subcube assignment satisfies.*

$$S_s \leq (5p\log_2 p + 5p + O(p^{1/2}))N^{1/2} + O(1) \tag{4.11}$$

Proof. The argument almost follows the previous one. We only need to notice the different number of processors at each step. We use the notation of Lemmas 1, 2 and 3 plus $p_{\ell x} = 4^{\ell/2}$ and $p_{\ell y} = 4^{\ell}$ denote the number of processors at each separator on the $\ell^{\text{th}}x$ and $\ell^{\text{th}}y$ levels, then we have the following estimates.

$$C_o^{1,y} \leq 2^d \left(2(p_{1y} - 1) \right) n_o \tag{4.12a}$$

$$C_o^{\ell,y} \leq 4^{d-\ell} \cdot 2 \left(2^{\ell}(p_{\ell y} - 1) \right) n_o, \quad \ell \geq 2 \tag{4.12b}$$

$$C_o^{1,x} \leq 2^{d+1}(p_{1x} - 1)n_o \tag{4.12c}$$

$$C_o^{\ell,x} \leq 2 \cdot 4^{d-\ell} \cdot 2 \left(2^{\ell-1}(p_{\ell x} - 1) \right) n_o, \quad \ell \geq 2 \tag{4.12d}$$

$$C_{ix}^{i+1,x} \leq 2 \cdot 2^{d-i} \left(2(p_{(i+1)x} - 1) \right) n_{ix} \tag{4.12e}$$

$$C_{ix}^{\ell,x} \leq 2 \cdot 4^{d-\ell} \cdot 2 \left(2^{\ell-i}(p_{\ell x} - 1) \right) n_{ix}, \quad \ell \geq i + 2 \tag{4.12f}$$

$$C_{ix}^{i,y} \leq 2 \cdot 2^{d-i}(p_{iy} - 1)n_{ix} \tag{4.12g}$$

$$C_{ix}^{\ell,y} \leq 4^{d-\ell} \cdot 2 \left(2^{\ell-i}(p_{\ell y} - 1) \right) n_{ix}, \quad \ell \geq i + 1 \tag{4.12h}$$

$$C_{iy}^{i+1,x} \leq 2 \cdot 2^{d-i} \left(2(p_{(i+1)x} - 1) \right) n_{iy} \tag{4.12i}$$

$$C_{iy}^{\ell,x} \le 2 \cdot 4^{d-\ell} \cdot 2 \left(2^{\ell-i-1}(p_{\ell x} - 1) \right) n_{iy}, \quad \ell \ge i+2 \tag{4.12j}$$

$$C_{iy}^{i+1,y} \le 2 \cdot 2^{d-i-1} \left(2(p_{(i+1)y} - 1) \right) n_{iy} \tag{4.12k}$$

$$C_{iy}^{\ell,y} \le 4^{d-\ell} \cdot 2 \left(2^{\ell-i}(p_{\ell y} - 1) \right) n_{iy}, \quad \ell \ge \ell+2 \quad . \tag{4.12l}$$

Combining (4.11) with (4.3), (4.6), (4.7c) and (4.9) and performing simplifications similar to those in the proof of Theorem 1, we obtain (4.11). This completes the proof.

As a byproduct of these estimates, we can bound the traffic volume for both assignments. Specifically, let V_g and V_s denote the total communication volume for the grid based subtree-subcube assignment and the standard subtree-subcube assignment, respectively.

Theorem 3. *Suppose $d > 3$, then we have*

$$V_g \le \left(14.4p - 3\log_2 p + O(1) \right) N + O(N^{1/2}) \tag{4.13}$$

$$V_s \le \left(31.85p + 64p^{1/2} + O(1) \right) N + O(N^{1/2}) \quad . \tag{4.14}$$

Proof. We only need to substitute n_o, n_{ix}, n_{iy} by $\frac{3}{2}n_o^2$, $\frac{13}{2}n_{ix}^2$, $\frac{9}{2}n_{iy}^2$, respectively, in the previous argument. To see this, let us first look at the level 0. Communication is required when the elimination reaches the B_{ij} unknowns. The message length (the length of the multiplier vector to be communicated) at each step is equal to the number of unknowns connected to the current unknown which is the number of uneliminated unknowns on B_{ij} and $\partial\,\Omega_{ij}$. Denote the message length associated with eliminating the m^{th} unknown on B_{ij} by ℓ_m. This length ℓ_m is bound by

$$\ell_m \le [4(k-1) + 4(k+1)] - m, \qquad m = 1, 2, \ldots, n \quad . \tag{4.15a}$$

with $n_o = 4(k-1)$. The total message volume is then the sum of the number of messages sent (start ups) times the lengths. The number of start ups is given by equations (4.2) and (4.12), the coefficients of n_o then are the coefficients of the ℓ_m. To count the total volume we have

$$\sum_{m=1}^{n_o} \ell_m \le \frac{3}{2}n_0^2 + \frac{15}{2}n_o \tag{4.15b}$$

and this replaces the n_o term in the expressions for start ups to give the traffic volume on level 0.

Similarly, we have, for the $i^{\text{th}}x$ level,

$$\ell_m \le (7n_{ix} + 6) - m, \qquad m = 1, 2, \ldots, n_{ix} \tag{4.16}$$

$$\sum_{m=1}^{n_{ix}} \ell_m \le \frac{13}{2} n_{ix}^2 + \frac{11}{2} n_{ix} \tag{4.17}$$

and for the $i^{\text{th}}y$ level,

$$\ell_m \le (5n_{iy} + 4) - m, \qquad m = 1, 2, \ldots, n_{iy} \tag{4.18a}$$

$$\sum_{m=1}^{n_{iy}} \ell_m \le \frac{9}{2} n_{iy}^2 + \frac{7}{2} n_{iy} \tag{4.18b}$$

Substituting these values in the estimates for the number of start ups gives the estimates of the traffic volume and completes the proof.

From Theorem 3 we see that the bound on V_s is about twice the bound on V_g. Both have the same optimal order $O(pN)$ as seen by examining the elimination at node 1. For illustration we give Table 4.1 which shows some values from these estimates with different p values. We see that the ratio of start ups between the two assignments grows gradually with p, with values between 3 and 5 for currently available hypercubes. The ratio of communication traffic volume stays close to 2.5. While these estimates are rather accurate for our model of hypercube communication, we must wait for actual measured values to be confident of the relative merits of these two assignments.

Table 4.1. *Example values for p= 64, 256 and 1024. The communication start up estimates S_g and S_s are from Theorems 1 and 2, the communication volume estimates are from Theorem 3.*

p	d	$\log_2 p$	S_g	S_s	S_s/S_g	V_g	V_s	V_s/V_g
64	3	6	$811N^{1/2}$	$2240N^{1/2}$	2.8	904N	2550N	2.8
256	4	8	$3243N^{1/2}$	$11520N^{1/2}$	3.6	3662N	9178N	2.5
1024	5	10	$12971N^{1/2}$	$56320N^{1/2}$	4.3	14716N	34662N	2.4

V. CONCLUSION

We propose a grid based subtree-subcube assignment strategy appropriate for solving PDE problems on hypercubes. The change from the standard subtree-subcube assignment is motivated by the multifrontal methods used in PDE solvers. For both assignments we give a complexity analysis for the communication costs of the

elimination considering both the total traffic volume and the number of start ups. The analysis is given for a PDE problem on a rectangular domain but the arguments can be extended to more general domains. The analysis of the traffic volume here is different from the recursive strategy used in [George, Lin and Ng, 1987] but the estimate obtained is of the same order. We show that our assignment has, theoretically, about half the total communication volume as the standard assignment and the number of communication start ups is improved by a factor of $38/(15 \log_2 p)$. Our assignment achieves the optimal order in traffic volume and number of start ups. It also provides load balancing and exploits fully the parallelism inherent in the problem. The effect of the communication reduction on practical execution may be dramatically larger than that expected theoretically because higher communication requirements may cause more unpredictable waiting during execution.

REFERENCES

Geist, G.A. and C.H. Romine (1988), "LU Factorization Algorithms on Distributed-Memory Multiprocessor Architectures", *SIAM J. Sci. Stat. Comput.*, **9**, pp. 639-649.

George, A., M. Heath, J. Liu, and E. Ng (1988), "Sparse Cholesky factorization on a local-memory multiprocessor", *SIAM Sci. Stat. Comput.*, **9**, 327-340.

George, A., J. Liu, and E. Ng (1987), "Communication reduction in parallel sparse Cholesky factorization on a hypercube", *Hypercube Multiprocessors* (M. Heath, ed.), SIAM Publications, Philadelphia, PA, 576-586.

Mu, M. and J.R. Rice (1989), "LU factorization and elimination for sparse matrices on hypercubes", to appear in the *Proc. of the Fourth Conference on Hypercube Concurrent Computers and Applications*, Monterey, CA.