SIAM J. COMPUT.
Vol. 21, No. 1, pp. 151–162, February 1992

012

EFFICIENT EMBEDDINGS OF TREES IN HYPERCUBES*

SANDEEP N. BHATT†, FAN R. K. CHUNG‡, F. THOMSON LEIGHTON§, AND ARNOLD L. ROSENBERG¶

Abstract. The boolean hypercube is a particularly versatile network for parallel computing. It is well known that multidimensional grid machines can be simulated on a hypercube with no communications overhead. In this paper it is shown that every bounded-degree tree can be simulated on the hypercube with constant communications overhead. In fact, the proof shows that every bounded-degree graph with an $O(1)$-separator can be embedded in a hypercube of the same size with dilation and congestion both $O(1)$. It is also proved that not all bounded-degree graphs can be efficiently embedded within the hypercube.

Key words. graph embedding, binary trees, boolean hypercube, dilation, expansion, congestion, tree decomposition

AMS(MOS) subject classifications. 68M10, 68Q10, 68R05, 68R10

1. Introduction.

The binary hypercube is emerging as one of the most popular network architectures for parallel machines. This is due partly to the facts that the hypercube has a simple recursive structure and that there are simple algorithms for message routing on the hypercube that work well in practice.

Another important consideration in the choice of network architecture is its ability to accommodate different algorithms efficiently. The problem of efficiently implementing various algorithms on parallel architectures has traditionally been studied as the "logical mapping problem" [2], [10] so that the problem of implementation becomes one of embedding the "data-dependency graph" underlying an algorithm within the processor interconnection graph. Many structured algorithms such as those in linear algebra [20] or the FFT algorithm [17] can be efficiently mapped onto the hypercube with minimal communication overhead. As an example, the N-node hypercube contains every N-node multidimensional grid, each of whose sides is a power of 2, as a subgraph. Hence grid-based algorithms can be executed efficiently on hypercubes.

In this paper we examine the ability of the hypercube to accommodate divide-and-conquer algorithms whose underlying structures are bounded-degree trees. Our main result is that the N-node hypercube can emulate every N-node bounded-degree tree with only a constant factor slowdown. In particular, we show how to embed any N-node bounded-degree tree within an N-node hypercube so that:

* Received by the editors December 19, 1990; accepted for publication January 28, 1991. This research was supported in part by Bell Communications Research.

† Department of Computer Science, Yale University, New Haven, Connecticut 06520. This author's research was supported by National Science Foundation grants MIPS-86-01885 and CCR-88-07426, by Defense Advanced Research Projects Agency grant CCR-89-08285, and by Air Force grant AFOSR-89-0382.

‡ Bell Communications Research, Morristown, New Jersey 07960.

§ Department of Mathematics and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. This author's research was supported by Air Force Office of Scientific Research grants AFOSR-86-0078 and AFOSR-89-0271, Defense Advanced Research Projects Agency grants N00014-80-C-0622 and N00014-89-J-1988, Army grant DAAL-03-86-K-0171, and a National Science Foundation Presidential Young Investigator Award with matching funds from Xerox and IBM.

¶ Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003. This author's research was supported by National Science Foundation grants DCI-85-04308, DCI-87-96236, and CCR-88-12567.

1. The mapping of tree nodes to hypercube nodes is one-to-one (i.e., the *max-load* of the embedding is one).
2. Each tree edge is mapped onto a hypercube path of length $O(1)$ (i.e., the *dilation* of the embedding is constant). The dilation of an embedding is a lower bound on communication delay, measured by the number of links a message must traverse.
3. Each hypercube edge is used to route only $O(1)$ tree edges (i.e., the *congestion* of the embedding is constant). Congestion bounds message-throughput rates, and thereby communication delay and queue sizes for holding messages in transit.
4. Only $O(1)$ tree edges are routed through each hypercube node (i.e., the *node-congestion* of the embedding is constant). Node-congestion bounds the total queue-size required at each node to hold messages in transit.

In other words, every tree can be embedded into a hypercube with expansion 1, and every other resource bounded by a constant. The embedding uses a divide-and-conquer approach involving multicolor separator theorems for binary trees, and is reminiscent of earlier work [8] on embedding graphs in grids for VLSI layout. Our techniques in this paper consequently translate into efficient embeddings (all resources bounded by a constant) for all bounded-degree graphs with $O(1)$-separators. Bounded-degree trees fall within this class, as do bounded-degree outerplanar graphs.

The paper is organized as follows. Section 2 summarizes related results; §3 presents a simple lower bound for embedding random trivalent graphs in hypercubes; §4 presents the basic technique for decomposing binary trees; §5 presents the final embedding. Section 6 concludes with some open questions.

2. Related results.

2.1. Previous work. The hypercube is known to contain, or nearly contain, many other natural structures as subgraphs. For example, the $n_1 \times n_2 \times \cdots \times n_k$ grid is a subgraph of the $\sum_{i=1}^{k} \lceil \log n_i \rceil$-dimensional hypercube.[1] Curiously, it is not a subgraph of any smaller hypercube. For example, a 3×5 grid is a subgraph of the 32-node hypercube, but it is not a subgraph of the 16-node hypercube. However, Chan [11] has recently shown that every $n_1 \times n_2$ grid can be embedded in a $\lceil \log n_1 n_2 \rceil$-dimensional hypercube with dilation no more than 2, and has further shown that the $n_1 \times \cdots \times n_k$ grid can be embedded one-to-one in the $\lceil \log n_1 n_2 \cdots n_k \rceil$-dimensional hypercube with dilation $O(k)$ [12].

The $(N-1)$-node complete binary tree is not a subgraph of the N-node hypercube,[2] although it can be embedded with dilation 2 since the N-node two-rooted complete binary tree (see Fig. 1) is a subgraph of the N-node hypercube [7], [18]. As a consequence, the $(N-1)$-node complete binary tree is a subgraph of the $2N$-node hypercube [30].

Even more complex and computationally powerful structures can be efficiently embedded within the hypercube. For example:

1. Leighton [22] showed that meshes of two-rooted trees are subgraphs of the hypercube; and Efe [13] showed that the mesh-of-trees is a subgraph of the hypercube;

[1] In this paper all logarithms are base 2.

[2] Both the tree and the hypercube are bipartite graphs. While the bipartite node sets for the hypercube are equal in size, they differ by a factor of 2 in the complete binary tree.

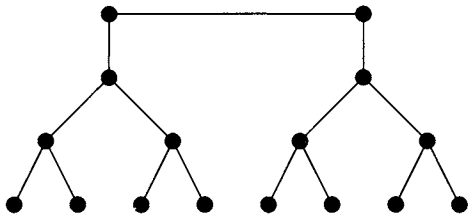

FIG. 1. *The two-rooted complete binary tree.*

2. Stout [28] showed that pyramid graphs can be embedded with dilation 2 and minimal expansion in the hypercube. Ho and Johnsson [19] improved this result to dilation 2 and congestion 2; and
3. Greenberg, Heath, and Rosenberg [17] showed that every FFT network is a subgraph of the smallest hypercube that can contain it. They also showed that the same result holds for butterfly networks of even order; butterflies of odd order can be embedded with dilation 2 in the smallest hypercube that can contain them.

In fact, the only bounded-degree graphs known not to have efficient embeddings in the hypercube are random graphs and expander-based graphs. In §4 we show that any constant-expansion, one-to-one embedding of such a graph into the hypercube must have dilation $\Omega(\log N)$, the worst possible (to within constant factors) since the N-node hypercube has diameter $\log N$.

We conjecture that the shuffle-exchange and deBruijn graphs are examples of bounded-degree graphs that cannot be embedded one-to-one into the hypercube with constant dilation and expansion. We know of no lower bounds for either case. We do not even know whether or not every bounded-degree planar graph can be embedded with constant dilation and expansion.

2.2. Extensions and subsequent work. This paper extends the results communicated earlier by the authors [5] when the node-congestion was not known to be $O(1)$. Our earlier communication [5] has since led to developments along a number of different directions, some of which we mention below.

Building on an earlier version [5], Monien and Sudborough [27] claim to have reduced the constants for dilation over our construction, but do not consider either node- or edge-congestion. For example, they claim that every N-node binary tree can be embedded with expansion 1 and dilation 5, or with dilation 3 and expansion $O(1)$. Whether or not any binary tree actually requires dilation 3 is open. Wagner [29] showed that every N-node binary tree is a subgraph of the hypercube with $O(N \log N)$ nodes. It is not known if every binary tree is a subgraph of an $O(N)$-node hypercube. Mayr [26] examined parallel algorithms which efficiently compute our embeddings.

The techniques in this paper were extended to embeddings within the butterfly, and related, networks. For example, the N-node complete binary tree can be embedded with constant expansion and dilation within the butterfly network [4] and, as a consequence of results in this paper, every N-node binary tree can be embedded one-to-one with constant expansion and $O(\log \log N)$ dilation and congestion within the butterfly network.

In [6] the authors applied the techniques used in this paper to construct a bounded-degree N-node graph that contains all N-node binary trees as spanning subgraphs.

This provided the first known example of a bounded-degree graph which is *universal* for all binary trees. Using an entirely different approach, Friedman and Pippenger [14] proved that every N-node bounded-degree tree is a (not necessarily spanning) subgraph of an $O(N)$-node expander graph.

Leighton and Malitz [23] constructed examples of N-node graphs with $O(N^\alpha)$-separators for which every one-to-one constant-expansion embedding into the hypercube must have dilation at least $\Omega(\alpha \log N / - \log \alpha)$. Thus, for example, there exist graphs with $O(\sqrt{N})$-separators for which every constant-expansion one-to-one embedding must have dilation at least $\Omega(\log N)$. This implies that embedding strategies based on separators alone cannot be used to obtain constant-dilation embeddings for planar graphs even if such embeddings exist.

Bhatt and Cai [3] considered the problem of maintaining dynamically evolving trees on the hypercube. They showed, among other results, that a simple randomized embedding technique guarantees dilation $O(\log \log N)$ and, with high probability, $O(1)$ max-load to maintain an N-node dynamic tree on the N-node hypercube. Leighton, Newman, Ranade, and Schwabe [24] improved this result to dynamic embeddings with $O(1)$ dilation, node-congestion, and edge-congestion while maintaining load balance.

The results mentioned above are all concerned with embedding bounded-degree graphs within hypercubes. For such graphs, embeddings with constant dilation and congestion utilize only $O(N)$ out of the $\frac{1}{2}N \log N$ communication edges of the hypercube. Greenberg and Bhatt [16] and Aiello, Leighton, Maggs, and Newman [1] extend these embeddings to multiple-path embeddings in which each edge of the host graph is mapped to multiple, edge-disjoint, short paths in the hypercube.

Along a different direction, Koch et al. [21] extend the study of graph embeddings to examine work-preserving emulations among different interconnection networks.

3. Definitions. Before proceeding with our results, we need a few definitions. An *embedding* $\langle \phi, \rho \rangle$ of a graph $G = (V_G, E_G)$ into a graph $H = (V_H, E_H)$ is defined by a mapping ϕ from V_G to V_H, together with a mapping ρ that maps each edge $(u, v) \in E_G$ onto a path $\rho(\phi(u), \phi(v))$ in H that connects $\phi(u)$ and $\phi(v)$. The *load* on a node $v \in H$ is the number of nodes of G that are mapped onto v; the *max-load* of an embedding is the maximum load over all nodes of H. The *expansion* of an embedding is the ratio of the size of V_H to the size of V_G.

The *dilation* of the edge (u, v) under $\langle \phi, \rho \rangle$ equals the length of the path $\rho(\phi(u), \phi(v))$ in H; the *dilation* of an embedding $\langle \phi, \rho \rangle$ is the maximum dilation of an edge in G. The *congestion* of an edge e_H in H under $\langle \phi, \rho \rangle$ equals $|\{e \in E_G : \rho(e) \text{ contains } e_H\}|$; the *congestion* of an embedding $\langle \phi, \rho \rangle$ is the maximum congestion of any edge in H. The congestion of a node $v_H \in H$ under $\langle \phi, \rho \rangle$ equals $|\{e \in E_G : \rho(e) \text{ contains } v_H\}|$; the *node-congestion* of an embedding $\langle \phi, \rho \rangle$ is the maximum congestion of any node in H.

We are interested in efficient embeddings of one family $\mathcal{G}$ of graphs into another family $\mathcal{H}$ of graphs. The families $\mathcal{G}$ and $\mathcal{H}$ are parameterized by the number of nodes they contain. By an expansion $e(N)$, dilation $d(N)$ embedding of $\mathcal{G}$ into $\mathcal{H}$ we formally mean a set of embeddings where every N-node graph $G_N \in \mathcal{G}$ is embedded into $H_{Ne(N)} \in \mathcal{H}$ with dilation no greater than $d(N)$. Embeddings for which the expansion and dilation do not grow with N are of particular interest. In what follows, we always mean embeddings among families of graphs, although we do not always make that explicit.

4. A lower bound. Not all bounded-degree graphs can be embedded one-to-one within hypercubes with small dilation and expansion. In particular, in what follows we show that every constant-expansion, one-to-one embedding of expander graphs [25] within hypercubes must have dilation $\Omega(\log N)$, the worst possible. One property of the family of expander graphs is that the removal of any m nodes from an N-node expander graph, $m \leq N/2$, requires that at least αm edges be cut, where $\alpha > 0$ is a constant independent of N.

PROPOSITION 4.1. *Every constant-expansion, one-to-one embedding of the family of expander graphs into the family of hypercubes requires dilation* $\Omega(\log N)$.

Proof. In fact, we will prove that the average dilation grows as $\Omega(\log N)$. Consider an embedding of an N-node expander G in the $2^k N$-node hypercube, where k is a nonnegative constant. Partition the set of $k + \log N$ dimensions of the hypercube into $t = \lfloor (k + \log N)/(k+1) \rfloor$ subsets $S_i, 1 \leq i \leq t$, of $k+1$ dimensions each, and possibly one more set with fewer than $k+1$ dimensions.

We first count the number of times edges of G traverse hypercube edges that lie in one of the dimensions of $S_i, 1 \leq i \leq t$. The removal of hypercube edges in dimensions within S_i splits the hypercube into 2^{k+1} blocks, each with $N/2$ nodes. This also splits the nodes of G into 2^{k+1} blocks, one of which contains at least $N/2^{k+1}$ nodes, and at most $N/2$ nodes. Since G is an expander, this means that at least $\alpha N/2^{k+1}$ edges of G ($\alpha > 0$ is some constant independent of N) each traverse at least one dimension of S_i. This is true for all i so the total number of dimension traversals is at least $t\alpha N/2^{k+1} = \Omega(N \log N)$. Because the number of dimension traversals is a lower bound on the sum of dilations of all edges, we have that the average dilation, and hence the dilation, is $\Omega(\log N)$. □

5. Embedding binary trees in thistle trees. To embed an arbitrary N-node binary tree T within the hypercube, we proceed in two stages. In the first stage, T is decomposed and efficiently embedded within the N-node *thistle tree*. The next section gives efficient embeddings of thistle trees within hypercubes; this second stage induces an efficient embedding of T in the hypercube.

Before defining thistle trees, however, we first present results on tree decomposition. These results are based on combinatorial techniques developed previously for VLSI layout [8], [9] and for constructing universal graphs for trees [6]. In particular, we will use a minor variant of the decomposition lemma from [6]. As mentioned in [8], the decomposition can be obtained in time polynomial in the size of T.

We begin with the notion of k-color bisectors. Suppose that every node of a graph G is colored with one of k colors. Further, let S be a set of nodes of G whose removal partitions the remaining nodes into two equal (to within one) subsets, both containing equal (again, to within one) numbers of nodes of each color, and such that there is no edge in G connecting a node from one subset to the other. Such a set S is called a k-color bisector of G.[3] Finally, we note that every N-node binary forest has a k-color bisector of size less than $k \log N$ [8], [9].

The following lemma is fundamental to our result. In what follows, the *depth* of a node in a tree is defined to be the distance from the root to that node; the root is at depth 0. In an N-node complete binary tree, a node at depth d is said to be at *level* $\log N - d$; leaves are at level 1.

[3] An immediate consequence of the definition is that a k-color bisector S for G can be extended into a k-color bisector $S' \supseteq S$ of any larger size, by removing nodes of the same color one by one, and alternating between the two separated subsets.

LEMMA 5.1. *Every N-node binary tree T can be mapped (many-one) to the level-$(\log N - 1)$ complete binary tree C so that* (a) *exactly $6\log(N/2^t) + 18$ nodes of T are mapped onto a node of C at depth $t < \log N - 7$, and at most 60 nodes of T are mapped to any node at depth $t = \log N - 7$, and no nodes of T are mapped at greater depth, and* (b) *any two nodes adjacent in T are mapped to nodes at most distance 3 apart in C. Furthermore, for every node of C, the numbers of nodes of T embedded within its two subtrees differ by at most 1.*

Remark. Lemma 5.1 is almost identical to Lemma 1 in [6]; the main difference being that the "exactly" in condition (a) above is replaced by "at most" in [6]. The proof remains almost identical, with the difference that whenever the proof in [6] uses bisectors of smaller size than stated in condition (a), we invoke the previous footnote to extend the bisector to the required size. By counting the number of nodes of T mapped at different depths, one can show that nodes of C at depths 0 through $\log N - 8$ are filled exactly as required, and at depth $(\log N - 7)$ each remaining subgraph has less than 60 nodes. The details are straightforward and are left to the reader.

For our purposes we will need to modify the above embedding slightly. Suppose that each node of C at level i (depth $\log N - 1 - i$) has maximum capacity i; i.e., at most i nodes of T can be placed at a level-i node of C. The number of nodes of T placed at nodes of depth $\log N - 8$ or less of C by Lemma 5.1 exceeds their capacity. In contrast, nodes at depth $\log N - 7$ and greater in C are assigned fewer nodes of T than their capacity allows. The following lemma states that we can perturb the mapping of T slightly so that capacity constraints are satisfied at every node of C, and without greatly increasing the distances between nodes adjacent in T.

LEMMA 5.2. *Every N-node binary tree T can be mapped (many-one) to the level-$(\log N - 1)$ complete binary tree C so that* (a) *at most i nodes of T are mapped onto a level-i node of C, and* (b) *any two nodes u and v that are adjacent in T are mapped to nodes U and V in C whose least common ancestor is at most distance 8 from each of U and V.*

Proof sketch. Given the embedding of Lemma 5.1, at each node of C make an ordered list of the nodes of T that are embedded there. Starting with the root, we "push" excess nodes of T down to lower levels as follows: when a node b is ready for "pushing," we fill b to capacity with the appropriate number of the "leftmost" vertices in the ordered list of nodes currently at b. The remaining nodes on the list are divided into two equal (to within 1) sublists and appended at the left end of the ordered lists for the children, b_0 and b_1, of b. The nodes b_0 and b_1 are now ready to be "pushed."

To establish that the above procedure maps every node of T within C (i.e., that no nodes of T are pushed out of C), we show inductively that the total number of nodes of T assigned to the subtree rooted at any node of C cannot exceed the total capacity of the subtree. Initially, this is true at the root. Suppose this is true at node b when it is ready to be pushed. We claim that after b has been pushed, the inductive hypothesis holds at each of the subtrees rooted at b_0 and b_1. Before b is pushed, the number of nodes of T mapped in the subtrees rooted at b_0 and b_1 are equal (to within one). Furthermore, equal (to within one) numbers of nodes are pushed onto b_0 and b_1 when b is pushed. The result is that the total numbers of nodes of T within the two subtrees remains equal (to within one) after b has been pushed. Therefore, if the capacity of either b_0 or b_1 is violated after b is pushed, it must be the case that the capacity of b was violated before b was pushed; this contradicts the inductive

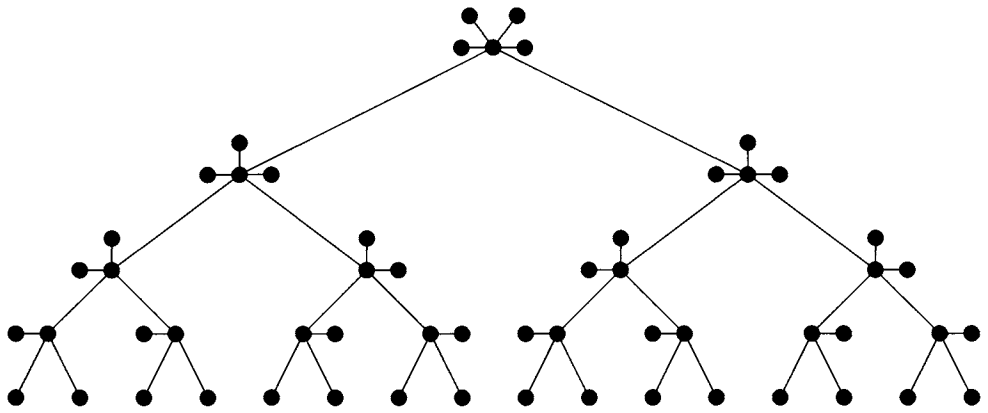

FIG. 2. *The thistle tree* T_5.

hypothesis.

For the second part of Lemma 5.2 we need to bound how far the push procedure can force a node of T to ripple down the tree. This is a simple calculation; the result is that the number of nodes assigned by Lemma 5.1 to depth at most ℓ in C is less than the total capacity of nodes with depth at most $\ell + 5$. This means that a node of T will be reassigned to a node of C with depth at most 5 greater than by Lemma 5.1. This suffices to guarantee part (b) of Lemma 5.2. □

5.1. Thistle trees. The decomposition of Lemma 5.2 motivates the definition of thistle trees. The thistle tree T_h is obtained by starting with a complete binary tree of $2^h - 1$ nodes and attaching to each level-i node, $1 \leq i \leq h$, $i - 1$ additional leaves called *thistles*. The thistle tree T_5 is shown in Fig. 2. The thistle tree T_k (of depth $k-1$) has $2^{k+1} - k - 2$ nodes. For convenience we assume that our binary trees have size $N = 2^{k+1} - k - 2$. This assumption will be removed later.

DEFINITION. If u is a thistle adjacent to node w, then we call w the *central node* of u, and denote $X(u) = w$. If u is not a thistle node, then it is a central node and we define $X(u) = u$.

6. Embeddings in the hypercube.

THEOREM 6.1. *Every N-node binary tree can be embedded one-to-one in a hypercube with expansion* 1, *dilation* $O(1)$, *and congestion* $O(1)$.

Proof. The decomposition of Lemma 5.2 is invoked to embed an arbitrary N-node binary tree one-to-one within a thistle tree. Map the nodes of T that are embedded within the same internal node in Lemma 5.2 onto distinct thistles adjacent in the thistle tree, with one node of T per thistle. This gives an embedding of T in the thistle tree with expansion 1 and max-load 1.

It remains to embed the N-node thistle tree within the hypercube. The next section investigates embeddings of thistle trees within hypercubes. We then complete the proof of Theorem 6.1 by showing that in the induced embedding both node- and edge-congestion are $O(1)$.

6.1. The inorder embedding of complete binary trees. There is a very natural way of embedding complete binary trees within hypercubes. As may be seen in Fig. 3, an inorder labeling of the 15-node tree yields an expansion-1, dilation-2

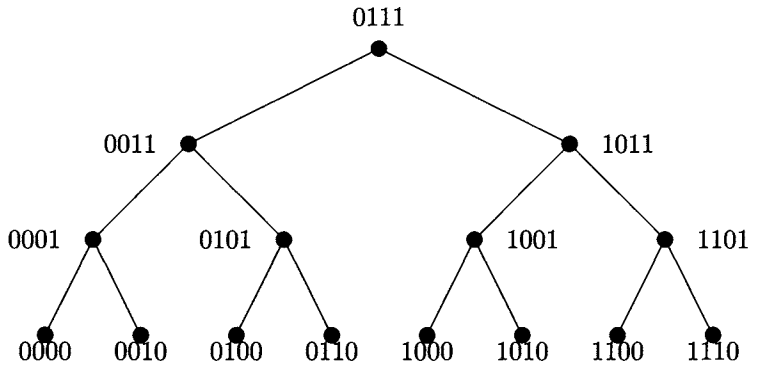

FIG. 3. *The inorder embedding of a complete binary tree.*

embedding. Each node v in the tree is labeled with the $\log N$-bit binary representation $b(v)$ of its inorder number; the hypercube address of v, $\phi(v)$, is defined to equal $b(v)$. This embedding scales to larger trees. In general, the ith leftmost node at level k ($k \geq 1, i \geq 0$) has inorder number $i2^k + 2^{k-1} - 1$; its left child (the $2i$th node at level $k-1$) has inorder number $i2^k + 2^{k-2} - 1$; and its right child has inorder number $i2^k + 2^{k-1} + 2^{k-2} - 1$. From this it follows that the inorder label of the left child of every node at level k differs from the inorder label of the node in bit position $k-2$ while the right child differs in bit positions $k-1$ and $k-2$. In other words, every left-edge (between a node and its left child) has dilation 1, and every right-edge has dilation 2.

We associate with each node u in the complete binary tree a path, $\tau(u)$, the *trace* of u, which starts at the left child of u and follows the rightmost path down the tree to a leaf. It is easily seen that every node w of the tree (except along the rightmost path from the root) lies in the trace of exactly one other node u. If a node w lies in $\tau(u)$, then we call u the *source* of node w.

The inorder numbering of a complete binary tree C of $2^n - 1$ nodes also has the following useful properties which can be verified in a straightforward manner.

1. The descendants of an internal node u that lie ℓ levels below u occupy an ℓ-dimensional subcube. The descendants that lie no more than ℓ levels below u reside in an $(\ell+1)$-dimensional subcube.
2. For every i, each node u at level i is adjacent in the hypercube to source(u) along an edge in dimension i. Therefore, the nodes in $\tau(u)$ are adjacent to u along dimensions $i-1, i-2, \cdots, 1$.
3. If S is the set of descendants of u that lie at most distance m away from u, and if u is at level i of C, then the set of nodes at level j ($j < i$) which are in the trace of nodes in S lie within an $(m+1)$-dimensional subcube. As j varies, these subcubes are disjoint but are defined by the same set of $m+1$ dimensions for all j.

6.2. Embedding the thistle tree. We embed a height-h thistle tree into a height-h complete binary tree as follows: embed the central node of each thistle onto its counterpart in the complete binary tree, and embed the $i-1$ thistles connected to a central node u at height i one-to-one onto the $i-1$ nodes in the trace $T(u)$. The properties of the inorder embedding mentioned in the previous section induce an

embedding of the height-h thistle tree into a 2^{h+1}-node hypercube with dilation 2, max-load 2, and expansion $\frac{1}{2}$. Of the two thistle-tree nodes mapped to one hypercube node, one is a central node and the other a thistle. We obtain a one-to-one embedding by first constructing a 2^{h+2}-node hypercube by taking two cubes of half the size. The entire thistle tree lies in one of the half-size cubes. We project each thistle node over to the corresponding empty hypercube node across the matching that connects the two half-size cubes to obtain an embedding with dilation 2, expansion 1, and max-load 1.

For convenience, we fix the following notation. Let V_T, V_{TT}, V_{CBT}, and V_H be, respectively, the node sets of the binary tree, thistle tree, complete binary tree, and the hypercube, so that $|V_T| = |V_{TT}| \leq |V_H| = N$, and $|V_{CBT}| = (N-1)/2$. The maps between these node sets are named as follows:

$$\begin{array}{lcll} \alpha & : & V_T & \mapsto & V_{TT}, \\ \beta & : & V_{TT} & \mapsto & V_{CBT} \text{ (not one-to-one)}, \\ \gamma & : & V_{TT} & \mapsto & V_H, \\ \phi & : & V_T & \mapsto & V_H \ (\phi = \gamma \circ \alpha). \end{array}$$

We now proceed with the proof of Theorem 6.1. In the embedding α, nodes u and v adjacent in T are mapped to nodes $\alpha(u)$ and $\alpha(v)$ whose central nodes are $X(\alpha(u))$ and $X(\alpha(v))$. The images $\gamma(X(\alpha(u)))$ and $\gamma(X(\alpha(v)))$ lie within a nine-dimensional subcube. How far apart can the images $\phi(u)$ and $\phi(v)$ be? In the worst case, $\phi(u) = \gamma(\alpha(u))$ and $\phi(v) = \gamma(\alpha(v))$ can be distance 1 away from $\gamma(X(\alpha(u)))$ and $\gamma(X(\alpha(v)))$, respectively, so the distance between $\phi(u)$ and $\phi(v)$ can be no greater than 11.

We have to find an assignment of paths within the hypercube to tree edges such that both node- and edge-congestion in the hypercube are $O(1)$. In the general case, suppose that we have to route tree edge (u, v) between hypercube nodes $\phi(u)$ and $\phi(v)$.

Let $U = \gamma(X(\alpha(u)))$ and $V = \gamma(X(\alpha(v)))$ denote the images of the central nodes of u and v. Let D_{UV} be the set of hypercube dimensions in which U and V differ. Further, suppose that $\beta(\alpha(u))$ and $\beta(\alpha(v))$ are at levels ℓ_u and ℓ_v of the complete binary tree, respectively, so that $\phi(u)$ and U differ in dimension ℓ_u, and $\phi(v)$ and V differ in dimension ℓ_v. Assume that $\ell_u < \ell_v$ (the case when they are equal is covered as a simpler subcase).[4]

The naive way (Fig. 4) to route edge (u, v) is to follow dimension ℓ_u from $\phi(u)$ to U, follow images of thistle-tree edges (within a nine-dimensional subcube) to V, and finally follow dimension ℓ_v to reach $\phi(v)$. The problem with this scheme is that the congestion along images of thistle-tree edges can be as large as $\Omega(\log N)$.

We can make both node- and edge-congestion $O(1)$ by traversing the dimensions in a different order, in three stages. As indicated in Fig. 5, in Stage 1 we follow a path which traverses dimensions in D_{UV} (in any order within the corresponding nine-dimensional subcube) from $\phi(u)$ to the node U_1. Observe that $\beta(\gamma^{-1}(\phi(u)))$ and $\beta(\gamma^{-1}(U_1))$ lie at the same level in the complete binary tree. In Stage 2 we follow dimension ℓ_v from U_1 to U_2, and in Stage 3 we follow dimension ℓ_u from U_2 to reach $\phi(v)$. Because $\ell_u < \ell_v$, it follows that $\beta(\gamma^{-1}(U_2))$ in the complete binary tree lies in the trace of $\beta(\alpha(v))$, and also that $\beta(\gamma^{-1}(U_2)), \beta(\gamma^{-1}(U_1))$, and $\beta(\alpha(u))$ all lie at level ℓ_u of the complete binary tree.

[4] The general case includes degenerate cases such as, for example, when $\alpha(u)$ or $\alpha(v)$ is a central node. We do not explicitly mention these cases.

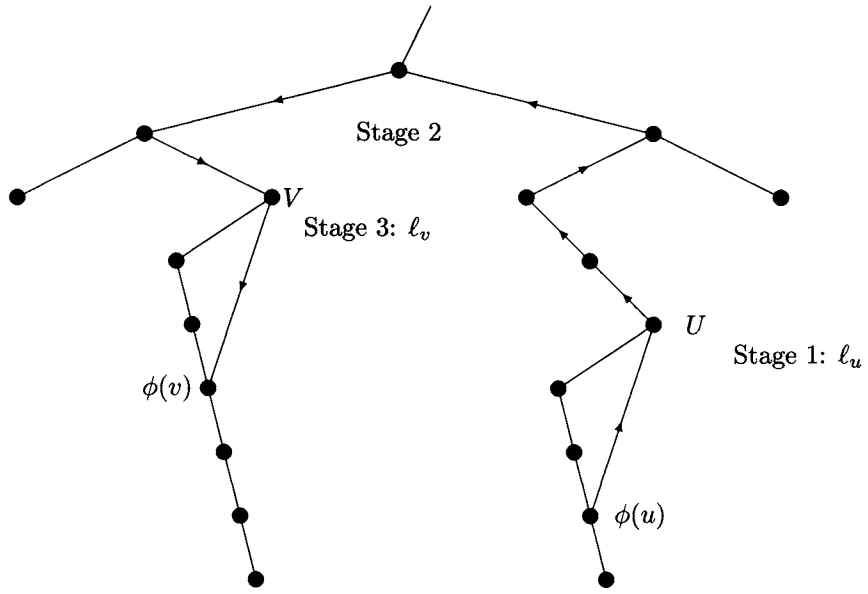

FIG. 4. *The naive scheme for routing paths.*

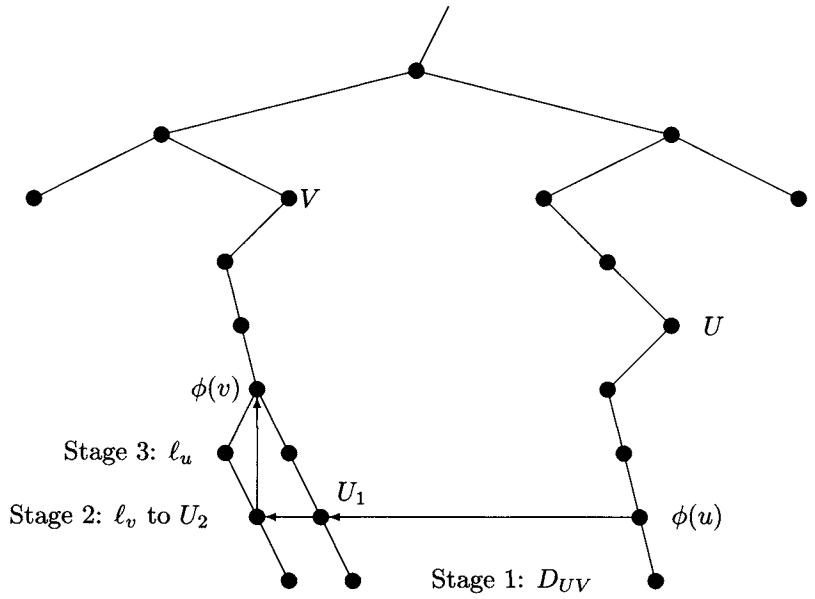

FIG. 5. *The modified scheme.*

With this modified routing, we claim that both node- and edge-congestion are $O(1)$. Consider the set of routes in Stage 1. By property 3 of inorder embeddings, the path from $\phi(u)$ to U_1 lies within a nine-dimensional subcube; the sources of their pre-images in the complete binary tree can be no further than eight levels away from their lowest common ancestor. Now u can be adjacent in T to another node w, in which case the route from $\phi(u)$ to W_1 in Stage 1 (define W, W_1, W_2 accordingly as for U, U_1, U_2) will lie within a nine-dimensional subcube defined by a set D_{UW}. The two sets D_{UV} and D_{UW} can be different so that the two routes starting at $\phi(u)$ can

lie in different nine-dimensional subcubes. However, every route starting at $\phi(u)$ is restricted to lie in one of only nine possible nine-dimensional subcubes because there are only eight choices for the lowest common ancestor, and each choice fixes a distinct nine-dimensional subcube. Because each such subcube has $O(1)$ nodes, and each node in a binary tree is the origin of up to three routes, both node- and edge-congestion in Stage 1 are $O(1)$.

Next consider Stage 2, in which each route is a single step. Because the node-congestion in Stage 1 is $O(1)$, we are guaranteed that each node in Stage 2 is the origin of only $O(1)$ messages. This suffices to guarantee that the edge-congestion in Stage 2 is $O(1)$. Observe that in Stage 3 all edges leaving nodes whose pre-images in the complete binary tree are at level ℓ_u are routed along dimension ℓ_u; therefore, at the end of Stage 2, routes terminating at U_2 correspond to edges incident to tree nodes which are mapped either onto U_2 or onto $\phi(v)$. Since each tree node has bounded degree, the node-congestion in Stage 2 is $O(1)$. In Stage 3 the node- and edge-congestion are $O(1)$ because of bounded-node degrees. Overall, therefore, both node- and edge-congestion are $O(1)$.

Finally, when the size N of the tree lies between $2^{k+1} - k - 2$ and 2^{k+1}, we first remove $N - 2^{k+1} + k + 2$ nodes, and embed the subtree of $2^{k+1} - k - 2$ nodes. Next, we use the fact [15] that within an m-dimensional hypercube m node-disjoint paths can be found to connect any m source nodes to any m sinks. By creating a sink which is vacant and sources wherever one of the removed nodes must be embedded, $k + 1$ additional nodes can be embedded by percolating nodes along the flow paths. This increases dilation, node-, and edge-congestion by $O(1)$. The last node can be inserted similarly, and the overall dilation, node-, and edge-congestion remain $O(1)$. □

7. Extensions and conclusions. This paper gives simulations of tree structures in the hypercube. The decomposition lemma (Lemma 5.2) for binary trees also provides optimal embeddings of binary trees within other networks. For example, we can show that every N-node binary tree can be embedded within an N-node complete binary tree with expansion 1 and dilation $O(\log \log N)$. By embedding a complete binary tree within the shuffle-exchange graph with expansion 1 and dilation 2, we obtain $O(\log \log N)$ dilation for arbitrary trees embedded within the shuffle-exchange graph. We have not yet determined whether or not these bounds are optimal to within constant factors.

All of our results on embeddings within the hypercube extend to arbitrary graphs of bounded degree with $O(1)$ separators. While our simulations are optimal to within constant factors, there is much room for reducing the overhead in expansion and dilation further. It would be satisfying to discover simpler ways to embed binary trees in the hypercube. For example, we do not know of any specific binary tree that cannot be embedded in the hypercube either with expansion 1 and dilation 2 or with expansion 2 and dilation 1.

Acknowledgments. Thanks to David Greenberg for helpful discussions and for his careful reading of the manuscript. Thanks also to Lennart Johnsson for early suggestions.

REFERENCES

[1] W. Aiello, F. Leighton, B. Maggs, and M. Newman, *Fast algorithms for bit-serial routing on a hypercube*, in Proc. Second Annual ACM Symposium on Parallel Algorithms and Architectures, 1990, pp. 55–62.

[2] F. BERMAN AND L. SNYDER, *On mapping parallel algorithms into parallel architectures*, J. Parallel Distrib. Computing, 4 (1987), pp. 439–458.

[3] S. BHATT AND J. CAI, *Take a walk, grow a tree*, in Proc. 29th Annual IEEE Symposium on Foundations of Computer Science, 1988, pp. 469–478.

[4] S. BHATT, F. CHUNG, J. HONG, F. LEIGHTON, B. OBRENIC, A. ROSENBERG, AND E. SCHWABE, *Optimal emulations by butterfly-like networks*, J. Assoc. Comput. Mach., to appear. (A preliminary version, *Optimal simulations by butterfly networks*, appears in Proc. 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 192–204.)

[5] S. BHATT, F. CHUNG, F. LEIGHTON, AND A. ROSENBERG, *Optimal simulations of tree machines*, in Proc. 27th Annual IEEE Symposium on Foundations of Computer Science, 1986, pp. 274–282.

[6] ———, *Universal graphs for bounded-degree trees and planar graphs*, SIAM J. Discrete Math., 2 (1989), pp. 145–155.

[7] S. BHATT AND I. IPSEN, *How to embed trees in hypercubes*, Tech. Report RR-443, Department of Computer Science, Yale University, New Haven, CT, 1985.

[8] S. BHATT AND F. LEIGHTON, *A framework for solving* VLSI *graph layout problems*, J. Comput. System Sci., 28 (1984), pp. 300–343.

[9] S. BHATT AND C. LEISERSON, *How to assemble tree machines*, in Advances in Computing Research 2, F. Preparata, ed., JAI Press, Greenwich, CT, 1984, pp. 95–114.

[10] S. BOKHARI, *On the mapping problem*, IEEE Trans. Comput., C-30 (1981), pp. 207–214.

[11] M. CHAN, *Dilation-2 embeddings of grids into hypercubes*, in Proc. IEEE Internat. Conference on Parallel Processing, 1988, pp. 295–298.

[12] ———, *Embedding of d-dimensional grids in optimal hypercubes*, in Proc. First ACM Symposium on Parallel Algorithms and Architectures, 1989, pp. 52–57.

[13] K. EFE, Personal communication, 1990.

[14] J. FRIEDMAN AND N. PIPPENGER, *Expanding graphs contain all small trees*, Combinatorica, 7 (1987), pp. 71–76.

[15] D. GREENBERG, *Minimum expansion embeddings of meshes in hypercubes*, Tech. Report RR-535, Department of Computer Science, Yale University, New Haven, CT, 1987.

[16] D. GREENBERG AND S. BHATT, *Routing multiple paths in hypercubes*, in Proc. Second ACM Symposium on Parallel Algorithms and Architectures, 1990, pp. 45–54.

[17] D. GREENBERG, L. HEATH, AND A. ROSENBERG, *Optimal embeddings of butterfly-like graphs in the hypercube*, Math. Systems Theory, 23 (1990), pp. 61–77.

[18] I. HAVEL AND P. LIEBL, *Embedding the polytomic tree into the n-cube*, Časopis Pěst. Mat., 98 (1973), pp. 307–314.

[19] C. HO AND S. JOHNSSON, *Embedding hyper-pyramids into hypercubes*, Tech. Report RR-667, Department of Computer Science, Yale University, New Haven, CT, 1988.

[20] S. JOHNSSON, *Communication efficient basic linear algebra computations on hypercube architectures*, J. Parallel Distrib. Comput., 4 (1987), pp. 133–172.

[21] R. KOCH, F. LEIGHTON, B. MAGGS, S. RAO, AND A. ROSENBERG, *Work-preserving emulations of fixed-connection networks*, in Proc. 21st Annual ACM Symposium on Theory of Computing, 1989, pp. 227–240.

[22] F. LEIGHTON, *Lecture notes on theory of parallel and distributed computation*, Tech. Report LCS/RSS/1, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1988.

[23] F. LEIGHTON AND S. MALITZ, *Separators and hypercube embeddings*, unpublished, 1990.

[24] F. LEIGHTON, M. NEWMAN, A. RANADE, AND E. SCHWABE, *Dynamic tree embeddings in hypercubes and butterflies*, in Proc. First ACM Symposium on Parallel Algorithms and Architectures, 1989, pp. 224–234.

[25] A. LUBOTZSKY, R. PHILIPS, AND P. SARNAK, *Ramanujan conjecture and explicit constructions of expanders*, in Proc. 18th Annual ACM Symposium on Theory of Computing, 1986, pp. 240–246.

[26] E. MAYR, Personal communication, 1987.

[27] B. MONIEN AND I. SUDBOROUGH, *Simulating binary trees on hypercubes*, in VLSI Algorithms and Architectures, Lectures Note in Computer Science 319, Springer-Verlag, Berlin, New York, 1988, pp. 170–180.

[28] Q. STOUT, *Hypercubes and pyramids*, in Pyramidal Systems for Computer Vision, V. Cantoni and S. Levialdi, eds., Springer-Verlag, Berlin, New York, 1986.

[29] A. WAGNER, *Embedding trees in the hypercube*, Tech. Report DCS/204-87, University of Toronto, 1987.

[30] A. Y. WU, *Embedding of tree networks into hypercubes*, J. Parallel and Distrib. Comput., 2 (1985), pp. 238–249.