Dense Edge-Disjoint Embedding of Complete Binary Trees in Interconnection Networks *

S. Ravindran, A. M. Gibbons and M. S. Paterson

Department of Computer Science, University of Warwick, Coventry CV4 7AL, England

Abstract

We describe dense edge-disjoint embeddings of the complete binary tree with n leaves in the following n-node communication networks: the hypercube, the de Bruijn and shuffle-exchange networks and the two-dimensional mesh. For the mesh and the shuffle-exchange graphs each edge is regarded as two parallel (or anti-parallel) edges. The embeddings have the following properties: paths of the tree are mapped onto edge-disjoint paths of the host graph and at most two tree nodes (just one of which is a leaf) are mapped onto each host node. We prove that the maximum distance from a leaf to the root of the tree is asymptotically as short as possible in all host graphs except in the case of the shuffle-exchange, in which case we conjecture that it is as short as possible. The embeddings facilitate efficient implementation of many P-RAM algorithms on these networks.

Keywords: parallel algorithms, graph embedding, binary tree, hypercube, de Bruijn network, shuffle-exchange network, mesh

1 Introduction

The P-RAM model of parallel computation is a shared memory model with constant-time access from any processor to any memory location. The constant-time memory access is not physically realisable in present-day hardware. When implementing algorithms for feasible models of parallel computation, a distributed memory machine consists of processors with local storage, where each processor is placed at a node of an *interconnection network*. It is natural to drawn upon the rich literature that has been developed for the P-RAM model over the last decade or so (see [4], for example). If we associate (in one-to-one correspondence) a P-RAM processor with each network processor, then a P-RAM algorithm can be implemented on the distributed memory machine in a time equivalent to the P-RAM complexity except that the constant time required for simultaneous (SIMD) memory accesses of the P-RAM processor is replaced by the time to solve an equivalent routing

*This work was partially supported by SERC grant GR/H/76487 and the ESPRIT II Basic Research Actions Programme of the EC under contract No.7141 (Project ALCOM II).

problem on the network. The most useful paradigm for such routing problems is the *permutation routing problem* in which each network processor sends and receives precisely one message of constant length. There are well-known algorithms ([9, 14, 7]) to solve the permutation routing problem in optimal $\Theta(d)$ time, where d is the diameter of the network, for the most commonly employed networks: hypercubes, meshes, de Bruijn and shuffle-exchange graphs.

The time complexities for implementations of the type just described can often be improved. For problems in NC, the running times are usually within logarithmic factors of the lower bound $\Omega(d)$ and this lower bound can often be attained. See for example [2, 5, 6]. Particularly effective strategies in this regard include the techniques of *compress-and-iterate* and the use of *graph embedding*. A widely applicable technique, advocated by Valiant [13] amongst others to obtain optimality in a different sense, is to use *parallel slackness* to hide the network *latency*. The optimality sought here is that of not exceeding the time-processor product (the *work* measure) of the P-RAM computation in the distributed memory machine implementation. If, for a particular architecture, the P-RAM can be emulated in this way then the P-RAM is said to be *universal* for that architecture. Valiant [13] has shown that the P-RAM is universal for the hypercubic architectures, for example, but it is not universal for constant degree networks [11] such as the mesh, de Bruijn or shuffle-exchange graphs. The emulation fails because of the limited bandwidth of the networks. On hypercubic networks, although the emulation is optimal in terms of the work measure, the computation is slowed down by a factor of $O(\log n)$.

Our concern in this paper is the improvement of running times for P-RAM algorithms when implemented on such interconnection networks. For the implementation of a large class of P-RAM computations (or subcomputations) on interconnection networks, the natural lower bound $\Omega(d)$ can often be attained by the use of certain strategies. If some algorithmic structure is used frequently then an embedding strategy could be usefully automated. Perhaps the most commonly occurring structure in this regard is the complete binary tree. It is precisely because such logarithmic depth structures are used (either explicitly or implicitly) that polylogarithmic time complexities are attained for many P-RAM algorithms. Thus many problems are placed in the complexity class NC, which is the class of efficiently solvable problems in this model.

In the P-RAM model, the complete binary tree is most usually employed as follows. Data for a problem (or subproblem) are placed at the leaves, and the required result is obtained by performing computations at the internal nodes in one or more sweeps up and down the tree, so that computations at the same depth are performed in parallel. It should be noted however that some algorithms may require simultaneous computation at an arbitrary number of nodes at different depths of the tree. If we are to embed the complete binary tree into the host topology of some distributed memory machine, we therefore need to satisfy the following requirements to achieve an *efficient* embedding:

1. *All tree nodes at the same depth should be mapped to disjoint host nodes if (as in the P-RAM computation) computations are to be performed in parallel at these nodes. In addition, P-RAM algorithms may require computation at nodes of the tree which*

are of different depths. Thus for greatest utility, the embedding should map at most a constant number of tree nodes to any node of the host graph.

2. *Tree edges at the same depth should correspond to edge-disjoint paths in the host graph if the commonest types of P-RAM algorithm using this technique are to be simulated. For greatest flexibility, all tree paths should be mapped to disjoint paths in the host graph.*

3. *The maximum distance (in terms of edges of the host graph) from the root to a leaf of the tree in the embedding should be minimised, in order that the routing time is minimised.*

4. *Consistent with satisfying the above points, the size of the host graph should be a minimum in the interests of processor economy.*

2 The embeddings

In the subsections that follow, we describe embeddings of complete binary trees with n leaves in hypercubes, de Bruijn graphs, *doubly-connected* two-dimensional meshes and shuffle-exchange graphs, each with n nodes. These are all topologies that have been advocated for interconnection networks and which we individually recall in the following subsections. By doubly-connected, we mean that each edge in the standard definition of the graph is replaced by two parallel edges. As we shall see, with the exception of the shuffle-exchange graph, the embeddings are such as to satisfy the following crucial properties which guarantee that in every respect the efficiency requirements stated in the previous section are met.

Embedding Properties:

1. Each node of the host graph is assigned exactly one leaf of the tree.

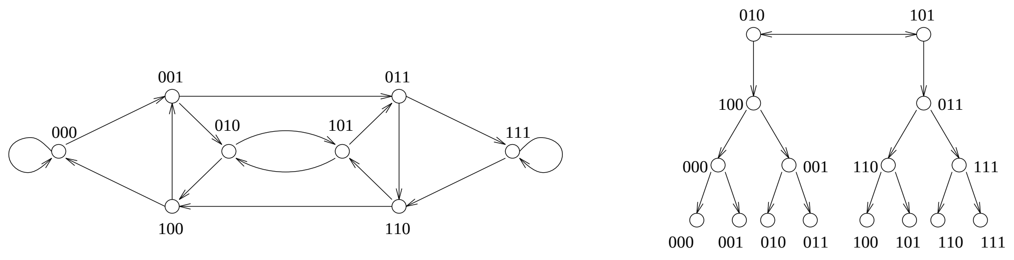
2. Each node of the host graph, except one, is also assigned exactly one internal node of the tree.

3. Distinct tree edges are mapped onto edge-disjoint (possibly null) paths in the host graph.

4. The maximum length of the images in the host graph of tree paths from a leaf to the root is as short as possible.

In the case of the shuffle-exchange graph, the embedding that we describe ensures that Embedding Properties 1–3 are satisfied. However, we can only conjecture that Property 4 is also satisfied by our embedding. In the embedding, the maximum length of an image of a leaf to root path is $2\log_2 n + 2$, whereas our best lower bound in this case is $(3/2)\log_2 n$.

Let DRCBT denote *Double Rooted Complete Binary Tree.* A DRCBT is a complete binary tree in which the path (of length 2) connecting the two children of the root is



(a) (b)

Figure 1.

replaced by a path, P, of length 3. Each of the two internal nodes of P (both of degree two) is a root of the DRCBT. These roots will be denoted by r_1 and r_2. In the hypercube, de Bruijn and shuffle-exchange graphs, each with n nodes, we shall in fact embed the DRCBT with $2n$ nodes (n leaves and n internal nodes).

The following subsections establish that Embedding Properties 1–3 hold for the embeddings described. We delay consideration of Embedding Property 4 until the next section. As we shall see, it is also the case that the *multiplicity* of the topologies (that is, the maximum number of parallel edges between any pair of nodes) is a minimum consistent with Embedding Properties 1–2.

2.1 Embedding in the de Bruijn graph

The *undirected de Bruijn graph* of degree m, $m \geq 0$, has $n = 2^m$ nodes which are named by all the distinct binary strings of length m. Each node $b_1 b_2 \cdots b_m$ is connected to node $b_2 b_3 \cdots b_m b_1$ by a *shuffle* edge, and to node $b_2 b_3 \cdots b_m \bar{b}_1$ by a *shuffle-exchange edge*. Here $\bar{b}_k$ is the complement of b_k. By implication, each node is also connected to $b_m b_1 b_2 \cdots b_{m-1}$ and to $\bar{b}_m b_1 b_2 \cdots b_{m-1}$.

For our purposes it is convenient to direct the edges of the de Bruijn graph from each node $b_1 b_2 \cdots b_m$ towards nodes $b_2 b_3 \cdots b_m b_1$ and $b_2 b_3 \cdots b_m \bar{b}_1$. Each node of the resulting directed graph has both out-degree and in-degree 2. Figure 1(a) shows the directed de Bruijn graph of degree 3.

It is also convenient here to direct the edges of the DRCBT. All edges are directed away from the roots (the edge between the roots will be bi-directed) as the example of Figure 1(b) illustrates. We say that each directed edge is from a parent to a child.

For $i \geq 0$, we inductively define directed graphs $G(i, m)$ as follows:

1. $G(0, m)$ consists of two isolated vertices each denoted by a binary string of length m consisting (for positive m) of alternating 0's and 1's, one string ending with a 0 and the other ending with a 1.

2. $G(i, m)$ is constructed from $G(i-1, m)$ as follows. From each vertex $v = b_1 b_2 \cdots b_m$ of $G(i-1, m)$ we add new directed edges (if they do not already exist) to (possibly

new) vertices $b_2b_3\dots b_m1$ and $b_2b_3\dots b_m0$. The former is called the *left-child* of v and the latter the *right-child* of v.

Lemma 1 *For $i < m$, $G(i,m)$ is a directed* DRCBT *and for $i = m$, $G(i,m)$ is a directed de Bruijn graph.*

Proof: First suppose that $i < m$ and, to avoid trivial cases, that $m > 2$. Let $[01]_k$ and $[10]_k$ denote the binary strings of length k consisting of alternating 0's and 1's that *end* with a 0 and a 1 respectively. Now, $G(1,m)$ is easily seen to be the directed DRCBT with four nodes. The roots are of the form $[01]_m$ and $[10]_m$, and are connected by anti-parallel edges. The two additional nodes are $c_1 = [01]_{m-2}00$, which is a right-child of one root, and $c_2 = [10]_{m-2}11$, which is the left-child of the other. As long as $i < m$ the inductive construction of $G(i,m)$ is such as to grow complete out-trees rooted at c_1 and c_2, each of depth i. To see this, it is sufficient to show that at each inductive step in which $G(i,m)$ is constructed from $G(i-1,m)$, the only new edges connect leaves of $G(i-1,m)$ to nodes whose labels are distinct from all previously obtained nodes and are distinct amongst themselves. Thus, the new nodes will be leaves of $G(i,m)$. It is easy to see that the new nodes are of the form $[10]_{m-1-i}11\alpha$ or $[01]_{m-1-i}00\alpha$, where α is a binary string of length $(i-1)$. These are distinct from all previous labels because, starting at the i^{th} position from the right, they contain either the substring 00 (if they are descendants of c_1) or the substring 11 (if they are descendants of c_2) and all previously existing nodes contain either 10 or 01 at this position. Any two of the new nodes descended from the same c_i will have different α's because each such α is uniquely determined by the path sequence of left or right edges that must be traced from c_i.

Thus we have proved that, for $i < m$, $G(i,m)$ is a directed DRCBT. By a trivial proof, if $i = m-1$ then the DRCBT has 2^m distinctly labelled nodes. Because this is the maximum number of distinct binary strings of length m, it follows that $G(m,m)$ will have the same set of nodes as $G(m-1,m)$. Also, every leaf of the $G(m-1,m)$ will have the form 00α or 11α, so that the rightmost substring of length $(m-1)$ is distinct amongst the labels of the leaves. This ensures that in the inductive construction of $G(m,m)$ from $G(m-1,m)$ the new edges (all directed from leaves of $G(m-1,m)$) will be directed to *distinct* nodes. In this way, every node of $G(m,m)$ has in-degree and out-degree 2. In fact, it trivially follows from the construction of $G(m,m)$ that every node $v = b_2\dots b_m$ is connected to $b_2b_3\dots b_m0$ and $b_2b_3\dots b_m1$ which are the children of v and edges are directed to v from $v_1 = 0b_1b_2\dots b_{m-1}$ and $v_2 = 1b_1b_2\dots b_{m-1}$. Of this last pair of nodes, if $b_1 = 0$ then v_1 is a leaf of $G(m-1,m)$ and v_2 is the parent of v in $G(m-1,m)$. If $b_1 = 1$ then the roles of v_1 and v_2 are reversed. Thus, the nodal connections of $G(m,m)$ are precisely those of the directed de Bruijn graph and this observation completes the proof. □

The following theorem follows trivially from the proof of the preceding lemma.

Theorem 1 *The directed* DRCBT *with n leaves can be embedded in the directed n-node de Bruijn graph so as to satisfy Embedding Properties 1–3.*

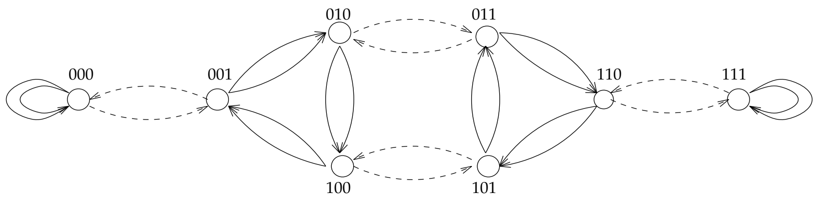


Figure 2.

Figure 1 provides an illustration of the theorem. Both (a) and (b) are $G(3,3)$. In (b) each node appears twice, once as a leaf of the directed DRCBT and once as an internal node. Copies of nodes are identified in (a) to show the directed de Bruijn graph.

2.2 Embedding in the shuffle-exchange graph

An undirected *shuffle-exchange* graph of dimension m has $n = 2^m$ nodes which are all the binary strings of length m. Each node $b_1b_2 \ldots b_{m-1}b_m$ is connected by an *exchange* edge to $b_1b_2 \ldots b_{m-1}\bar{b}_m$ and by a *shuffle* edge to $b_2b_3 \ldots b_m b_1$. By implication, each node $b_1b_2 \ldots b_m$ is also connected by a shuffle edge to $b_m b_1 b_2 \ldots b_{m-1}$.

Figure 2 shows the shuffle-exchange graph of degree 3 in which each exchange edges has been replaced by a pair (dashed for emphasis) of anti-parallel edges and each shuffle edge has been replaced by a pair of parallel edges. This particular form is derived from a previously known (see [8], for example) embedding of the de Bruijn graph in the shuffle-exchange graph. This embedding has both congestion and dilation of 2. The embedding is obtained by removing each shuffle-exchange edge $(b_1b_2 \ldots b_m, b_2b_3 \ldots b_m\bar{b}_1)$ from the directed de Bruijn graph and replacing it with the directed path consisting of the shuffle edge $(b_1b_2 \ldots b_m, b_2b_3 \ldots b_m b_1)$ followed by the exchange edge $(b_2b_3 \ldots b_m b_1, b_2b_3 \ldots b_m\bar{b}_1)$ of the shuffle-exchange graph. Because the graph now uses just the nodal connections of the shuffle-exchange graph, it is precisely such a graph but with parallel and anti-parallel edges. In this way, for example, it is easy to see that Figure 2 can be derived from Figure 1(a). The following theorem follows immediately from this embedding of the de Bruijn graph and from Theorem 1.

Theorem 2 *The* DRCBT *with n leaves can be embedded in the doubly-connected shuffle-exchange graph with n nodes so as to satisfy Embedding Properties 1–3.*

2.3 Embedding in the two-dimensional mesh

The two-dimensional doubly-connected mesh is the target graph for the embedding of this subsection. Adjacent nodes are connected by a pair of anti-parallel edges. The guest graph of the embedding is the complete binary in-tree, that is, a complete binary tree in which the edges are directed towards the root. We prove the following theorem [3].

Theorem 3 *For all $m \geq 1$, there are embeddings of the complete binary trees with 2^{2m} and 2^{2m+1} leaves into a doubly-connected $2^m \times 2^m$ mesh and a doubly-connected $2^m \times 2^{m+1}$*

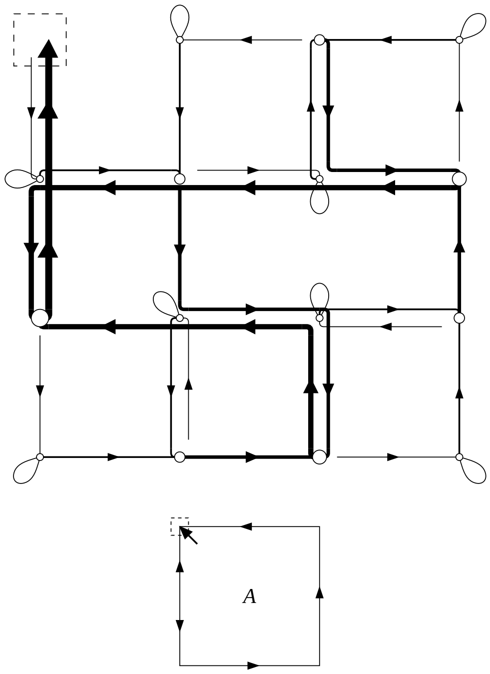


Figure 3.

mesh respectively, which satisfy Embedding Properties 1–3.

Proof: First consider the embedding in a square for the tree with 2^{2m} leaves. The case $m = 1$ is easy. For $m = 2$, Figure 3 shows one possible embedding in the 4×4 mesh. In this figure, the internal tree nodes and the the paths corresponding to the tree edges, are drawn with increasing size and boldness from leaves to root respectively. The edges are directed towards the root. The leaf nodes are not shown explicitly since there is one at each mesh node. Note that some tree edges, incident with the leaves, are mapped to null paths indicated by loops in the figure. The root is embedded on the left side, but the heavy path shown from this to the top-left corner is used later in larger embeddings. The node distinguished with a dotted square in the figure is that unique node which has not yet been assigned an internal tree node. The small diagram underneath gives the salient features of this embedding, A_2, for use in the recursive construction. The arrows on the perimeter indicate the usage so far of the outside edges, and show that all the clockwise outside edges on three of the sides are as yet unused. The construction requires also an alternative 4×4 embedding, B_2, shown in Figure 4. The root here is embedded in the interior of the square but there is an outgoing path from it to the lower-right corner. This time, all the clockwise edges on the top, left and bottom sides are free.

The next stage in our construction, the embeddings for $m \geq 3$, is shown in Figure 5. Three A_{m-1}'s and one B_{m-1} are combined to give embeddings of the 2^{2m}-leaf tree in a $2^m \times 2^m$ mesh. The three new internal tree nodes required are shown by white and black circles, and are connected by paths of appropriate weight. For the recursion, the embedding is continued in two different ways. The black root node can be connected to the top-left

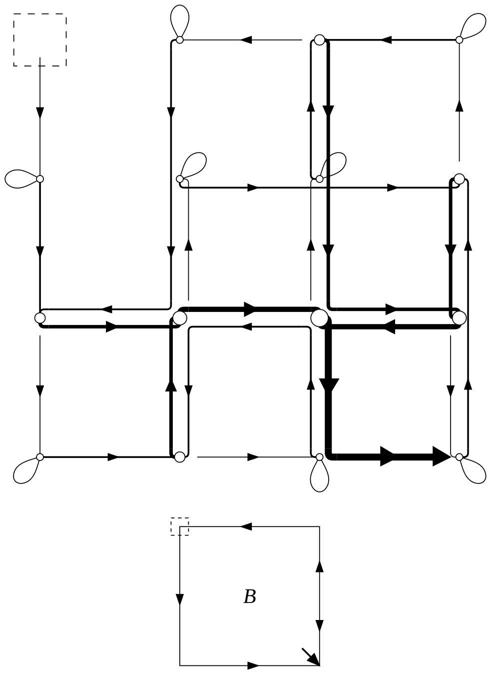


Figure 4.

corner by one of the shaded paths shown, or joined to the lower-right corner by another shaded path. The first alternative yields an embedding A_m which has edge characteristics of type A given by the small diagram in Figure 3, while the second similarly yields B_m. The arrangement shown in Figure 5 therefore represents a recursive step by which the construction can be continued indefinitely. The third shaded path illustrated to the lower-left corner, will be used in the 2^{2m+1}-leaf embedding. For the case $2^m \times 2^{m+1}$, if $m \geq 3$ we can connect seven copies of A_{m-1} with one copy of C_{m-1} as shown in Figure 6. The cases where $m < 3$ are simple. □

2.4 Embedding in the hypercube

A hypercube has n nodes (where $n = 2^m$, for some positive integer m) labeled from 0 to $n-1$ in binary and such that there is an edge between two nodes if and only if their binary labels differ in exactly one bit. For completeness, we briefly present the following result which was first described in [10].

Theorem 4 *The double-rooted complete binary tree with $n \geq 32$ leaves can be embedded with unit dilation in the hypercube with n nodes, so as to satisfy Embedding Properties 1–3.*

Proof: We construct the embedding recursively starting with the base case of $n = 32$, as shown in Figure 7. In the figure nodes occur at the corners of the squares defined by the dashed lines, and the labels of nodes in the top left-hand quarter of the figure are shown. The first two binary bits of the labels of nodes in the other quarters are shown at the center

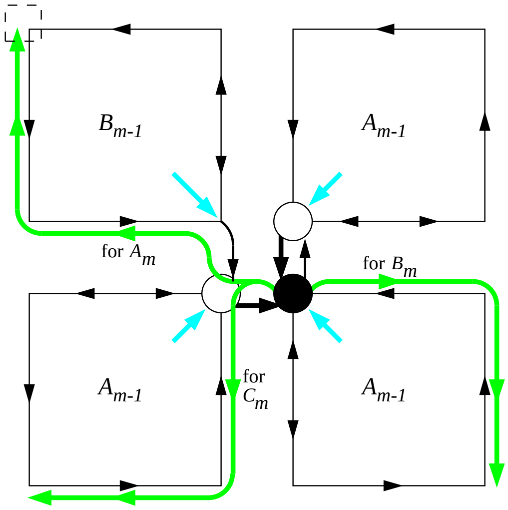


Figure 5.

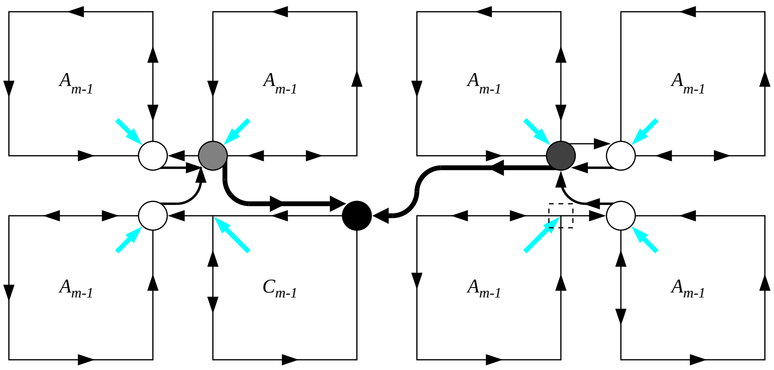
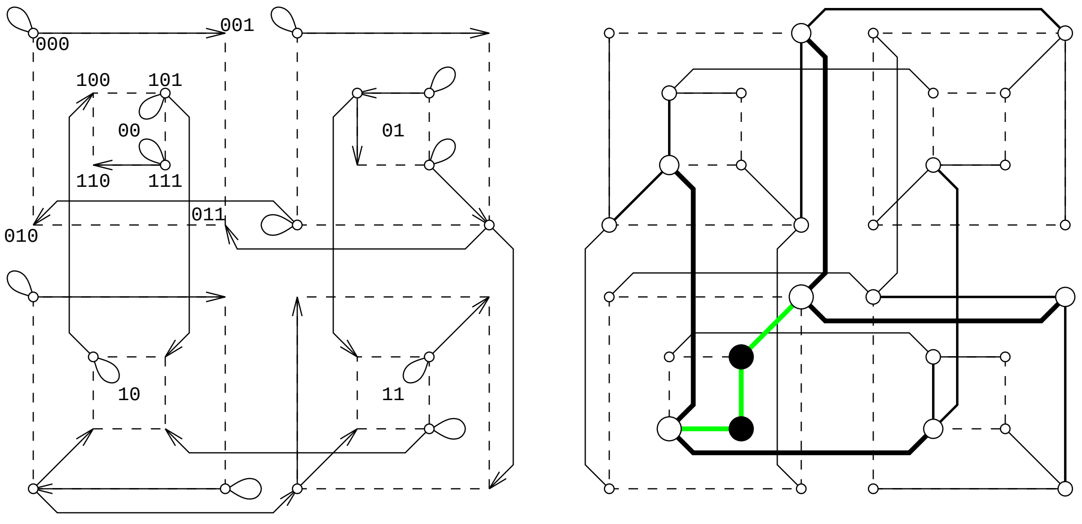
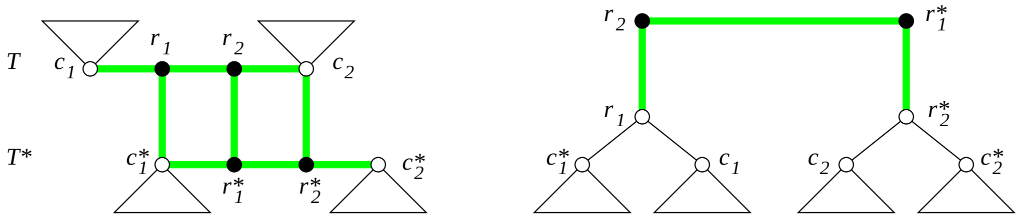


Figure 6.



(a) (b)

Figure 7.



(a) *(b)*

Figure 8.

of their quarter of the figure. The last three binary bits of such an address will be the same as the corresponding node in the top left-hand quarter. Generally speaking, figures will only show those edges of the hypercube that are of interest. Dashed edges correspond to certain hypercube edges but are used merely as an aid in locating nodes in the layout. For clarity, two figures (7(a) and (b)) are used to describe this case. Figure 7(a) shows the embedding of those tree edges which have leaves as end-points. For clarity, some embedded tree edges point towards that endpoint which is a leaf of the tree. Some tree edges are mapped to null paths which are indicated by loops. Figure 7(b) shows the embedding of all other tree edges. Notice that shaded edges are used for the path of length 3 on which full circles denote the roots of the embedded DRCBT. Also, notice that the three edges on this path belong to three different dimensions of the hypercube. In Figure 7(b), the internal nodes are drawn with increasing size and the tree edges are drawn with increasing boldness the nearer they are to the root. It is easy to see that this base case satisfies Properties 1–3 in all respects.

Figure 8 illustrates the inductive step in the construction of the embedding of the DRCBT with n leaves in the hypercube with n nodes from two embeddings of $n/2$-leaf

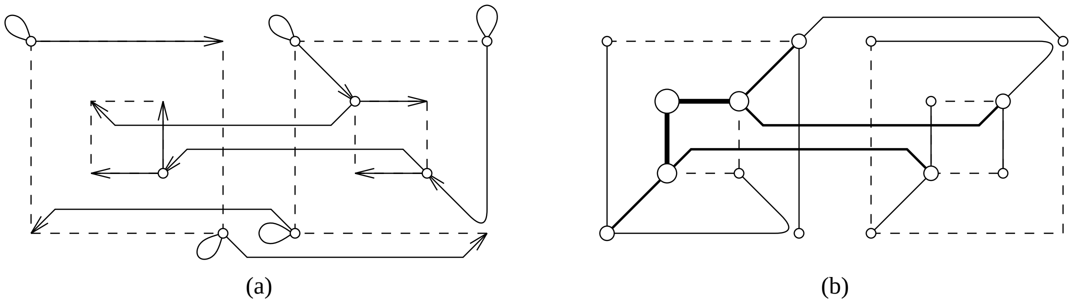


Figure 9.

DRCBT's in hypercubes with $n/2$ nodes. These two embeddings are denoted by T and T^* in Figure 8(a). The shaded vertical edges in that figure are edges of the new dimension of the constructed hypercube. The shaded horizontal paths ((c_1,r_1,r_2,c_2) and (c_1^*, r_1^*, r_2^*, c_2^*)) are the paths of length 3 containing the roots of the embedded DRCBT's with $n/2$ leaves. The triangular shapes attached to children of these possible roots represent the embedded subtrees rooted at these children. The two smaller hypercubes are oriented so that r_2 and r_1^* correspond, the dimensions of the edges (r_1,r_2) and (c_1^*,r_1^*) correspond, and the dimensions of the edges (r_2,c_2) and (r_1^*,r_2^*) correspond. This is always possible given the edge-transitivity of the hypercube and given that each of the horizontal shaded paths of length 3 has each edge of different dimension. Figure 8(b) shows the embedding of the DRCBT with n leaves and unit dilation in the constructed hypercube with n nodes. The labeling of nodes in this figure makes clear its derivation from Figure 8(a), and shows that the new central path of length 3 uses three different dimensions. □

For complete binary trees we have a similar theorem.

Theorem 5 *The complete binary tree with $n \geq 16$ leaves can be embedded with dilation two in the hypercube with n nodes, so as to satisfy Embedding Properties 1–3.*

Proof: For $n = 16$, an embedding satisfying Properties 1–3 is shown in Figure 9. The conventions used to represent the hypercube are similar to those of Figure 7. Again, for convenience of illustration the embedding of tree edges attached to leaves are shown in one diagram (Figure 9(a)) and the embeddings of all other edges in another (Figure 9(b)).

For $n \geq 32$, the embedding is derived directly from the embedding of the DRCBT given by Theorem 4. □

3 Depths of the embedded trees

In this section we examine the quality of our embeddings from the point of view of Embedding Property 4. The maximum distance from the root to a leaf in the image of the complete binary tree for any host graph is an important algorithmic parameter. It is a measure of the routing time required for a single sweep of the balanced binary tree. We denote this distance by $P(n)$ for the complete binary tree with n nodes. If the embedding satisfies Embedding Properties 1–3 of Section 2, the $O(\log n)$ routing time of the P-RAM algorithm for such a sweep translates to $O(P(n))$ for the interconnection network.

We first determine $P(n)$ for the embeddings in each of the four interconnection networks considered in this paper. Then we establish lower bounds for the maximum root-to-leaf distances for these embeddings, showing that, in all cases except for that of the shuffle-exchange graph, our values of $P(n)$ are asymptotically as short as possible. We conjecture that this is also true for the shuffle-exchange graph, although there is a gap between the $P(n)$ of our embedding and the lower bound obtained.

3.1 Maximum root-to-leaf distances of the embeddings

For the n-node hypercube (for $n \geq 32$) and for de Bruijn graphs, $P(n)$ is $\log_2 n + 1$. This is because each edge of the n-leaf DRCBT is mapped into at most one edge of the n-node host and there are root-to-leaf paths for which every such edge is mapped to precisely one edge of the host. Thus, for the embedded DRCBT the maximum length of root-to-leaf paths is $\log_2 n$ in these cases. This translates to $\log_2 n + 1$ for the complete binary tree when its root is identified with a particular one of the two roots of the DRCBT.

In our embedding of the n-leaf DRCBT in the doubly-connected n-node shuffle-exchange graph one of the pair of edges from each parent to its children is mapped into two edges of the host and so for this case $P(n) = 2(\log_2 n + 1) = 2\log_2 n + 2$.

We now consider the embedding of the 2^{2m}-leaf complete binary tree into a doubly-connected $2^m \times 2^m$ mesh. Let $D(m)$ be $P(n)$ when expressed as a function of m. It is a trivial matter to construct an embedding for $m = 1$ with $D(m) = 2$. For $m = 2$, we see by inspection of the embedding B_2 of Figure 4 that this maximum distance is 6 mesh steps and so $D(2) = 6$. For square meshes with $n = 2^{2m}$ nodes, $m \geq 2$, let A(m), B(m) and C(m) be the maximum distances from a leaf to the output from the top left corner of pattern A_m, the lower-right corner of pattern B_m and the lower-left corner of pattern C_m, respectively. From Figures 3 and 4, we see that A(2) = 10 and B(2) = 8. A corresponding layout C_2 with C(2) = 9 is easy to derive from B_2. We can verify from Figure 5 the following recurrence equations for $m \geq 3$:

$$\begin{aligned}
A(m) &= D(m) + 2^m, \\
B(m) &= D(m) + 2^m - 2, \\
C(m) &= D(m) + 2^m - 1, \\
D(m) &= max\{A(m-1) + 2, B(m-1) + 2\} \\
&= D(m-1) + 2^{m-1} + 2 \ .
\end{aligned}$$

The solution to these equations is:

$$D(m) \;=\; 2^m + 2m - 2 \text{ for } m \geq 1,$$

that is,

$$P(n) = \sqrt{n} + 2\log_2 n - 2 = \sqrt{n} + O(\log n) \ .$$

Finally, for the embedding of a 2^{2m+1}-leaf tree into a doubly-connected $2^m \times 2^{m+1}$ mesh, let $D'(m)$ be the corresponding maximal leaf-to-root distance. We may verify in Figure 6 that:

$$D'(m) = max\{A(m-1)+4, C(m-1)+3\} + 2^{m-1}$$

and so, in this case,

$$P(n) = \frac{3}{2}\sqrt{\frac{n}{2}} + O(\log n) \ .$$

3.2 Lower bounds for the embedded tree depths

Here we obtain lower bounds for the depth of the complete binary tree for the different embedding problems and show that, in the cases of the mesh, hypercube and de Bruijn graphs, these bounds asymptotically match the values of $P(n)$ that were obtained in the previous subsection.

For a given graph, let its *radius* ρ be the minimum distance r such that, for some *central* node c, every node is at a distance at most r from c. Clearly, for any embedding of a complete binary tree in a communication network in which Embedding Properties 1–3 are met, a lower bound for $P(n)$ is provided by ρ.

Lemma 2 *The following relationships hold for graphs with $n = 2^m$ nodes:*

(i) for the hypercube: $\rho = m$,

(ii) for the de Bruijn graph: $m - 1 \geq \rho \sim m$, and

(iii) for the shuffle-exchange graph: $\rho \sim \frac{3}{2}m$.

Proof: The result is trivial for the hypercube. For the *directed* de Bruijn graph $\rho = m$, but we can take advantage of the extra possibilities of the undirected graph. It is easy to show that the node $0^{m-1}1$ has distance at most $m-1$ from every node. We have verified that $\rho = m - 1$, for $2 \leq m \leq 9$, and we expect that $\rho \in \{m-2, m-1\}$ for all $m \geq 2$. The radius of the shuffle-exchange graph is more elusive. The nodes $[01]_m$ and $[10]_m$ demonstrate that $\rho \leq 3m/2 + 1$, but experimental results show that these nodes are not centres for $3 \leq m \leq 9$.

The lower bounds for the de Bruijn and shuffle-exchange graphs follow from simple counting arguments. We have only to estimate the numbers of nodes b_r reachable within a distance r of an arbitrary node. Our bounds follow from the requirement that $b_\rho = 2^m$. □

For the hypercube we obtain a marginally stronger lower bound for $P(n)$ in the following lemma.

Lemma 3 *For any leaf-disjoint embedding of a complete binary tree with $n = 2^m$ leaves into the n-node hypercube, $P(n) \geq m + 1$.*

Proof: If there were an embedding satisfying Embedding Properties 1–3 with $P(n) = m$, this would imply that a unit dilation embedding of the complete binary tree (perhaps with some parent-to-leaf edges mapped to null paths) was possible in the hypercube. The induced embedding of the subtree consisting of all internal tree nodes would be vertex-disjoint, with every edge being mapped to a hypercube edge, i.e., the subtree would be a subgraph of the hypercube. Both the subtree and the hypercube are bipartite graphs, and the embedding would respect the bipartition of the nodes. In the case of the hypercube both halves of the bipartition contain the same number of nodes, whereas for the subtree the two parts contain $\lfloor 2^m/3 \rfloor$ and $\lfloor 2^{m+1}/3 \rfloor$ nodes. This contradiction proves the lemma. □

Lemma 4 *For an arbitrary leaf-disjoint embedding of a complete binary tree with n leaves into the integer mesh graph $\mathbb{Z} \times \mathbb{Z}$,*

$$P(n) \geq \sqrt{\frac{n}{2}} - O(1) \ .$$

For such an embedding into an $r \times s$ rectangle, where $n = rs$,

$$P(n) \geq \rho = \left\lceil \frac{r-1}{2} \right\rceil + \left\lceil \frac{s-1}{2} \right\rceil = (r+s)/2 - O(1) \ .$$

Proof: The number b_r of vertices of $\mathbb{Z} \times \mathbb{Z}$ within distance r of the origin is given by: $b_r = 1 + \sum_{i=1}^{r} 4i = 2r(r+1) + 1$. For any injective mapping of n leaves into the mesh, if $n > b_r$ then some vertex has to be mapped to a mesh node at distance greater than r from the root. □

We summarise the above lemmas and our embedding results in the following theorem.

Theorem 6 *For any of the families of host interconnection graphs considered, let $p(n)$ be the minimal value of $P(n)$ achieved by any embedding of the n-leaf complete binary tree satisfying the Embedding Properties 1–3.*

(i) For the hypercube: $p(n) = \log_2 n + 1$ for $n \geq 16$.

(ii) For the undirected de Bruijn graph: $p(n) \sim \log_2 n$.

(iii) For the shuffle-exchange graph: $\frac{3}{2}\log_2 n - o(\log n) \leq p(n) \leq 2\log_2 n + O(1)$.

(iv) For the two-dimensional mesh: $p(n) \sim \sqrt{n}$ if $\log_2 n$ is even; $p(n) \sim \frac{3}{2}\sqrt{n}$ if $\log_2 n$ is odd.

This theorem shows that our embeddings are asymptotically optimal with respect to Embedding Property 4 for the hypercube, de Bruijn, and two-dimensional mesh interconnection networks. We conjecture that our embedding for the shuffle-exchange graph is also asymptotically optimal, although in this case the value of $P(n)$ exceeds our lower bound by a factor of $\frac{4}{3}$.

4 Further remarks and algorithmic issues

Here we briefly justify the use of parallel or anti-parallel edges in some of our embeddings. We then comment on the complexity gains afforded by our embeddings when used for P-RAM implementation on the associated interconnection networks.

A natural question to consider is whether the pairs of anti-parallel edges are necessary for the mesh. Can the complete (undirected) tree be densely embedded in the usual undirected mesh? Each mesh node (except two) is host to one leaf vertex with degree one and one internal vertex with degree three, and so has a total of at least four embedded edges incident with it. Note that some of the edges adjacent to leaves can be mapped into paths of length zero, the loops in our figures, and so some mesh nodes may require only *two* of their incident mesh edges. Thus there is no immediate contradiction from degree considerations. However, we now consider local details and easily find a contradiction. Consider boundary mesh nodes, away from the one special node that does not host an internal tree vertex. Any such node has degree less than four and so must have a loop in the embedding. It therefore is host to a leaf vertex and the internal node adjacent to that leaf, and requires one incoming path from another leaf, and one outgoing path to the parent vertex. Since the neighbouring boundary nodes are in the same predicament, there is an impossible situation at the boundary, even worse if it is at a corner.

It is also easy to see that we need parallel (or anti-parallel) edges for the shuffle-exchange graph when embedding the complete binary tree if the embedding is consistent with Embedding Properties 1–2. This is because the shuffle-exchange graph has degree 3 but any internal node of the tree which is not adjacent to a leaf has to be mapped to the same node of the shuffle-exchange graph as a leaf. This requires that at least four tree edges have this shuffle-exchange node as an end-point which is not possible without parallel (or anti-parallel) edges being added to the shuffle exchange graph to ensure edge-disjointness of the embedding. Notice that it also follows that the dilation of the embedding must be greater than 1.

For the hypercube, de Bruijn and shuffle-exchange graphs our embeddings show that the complete binary tree can be embedded with disjoint edges in hosts that are generally half the size compared with previously described embeddings, without detriment to the time complexities of P-RAM algorithms that use complete binary trees. The embedding of a complete binary tree in the hypercube described in [1] meets all our efficiency requirements except that the host graph is twice as large as it need be: the n-leaf complete binary tree is embedded in the hypercube with $2n$ nodes. In [7] (pages 407-410), an embedding is described in which the n-leaf tree is embedded in the n-node hypercube. However, in this embedding, up to $\log_2 n$ tree nodes of different depths are mapped to a single node of the hypercube. Although the embedding is such as to facilitate the efficient implementation of most P-RAM algorithms, there may be difficulties in the exceptional cases when simultaneous computation is required to take place at an arbitrary number of different levels within the tree. Such an example is cited later in this section.

For the two-dimensional mesh, our embeddings may not only reduce the size of the host graph but will also improve running times of the implementations. For example, in the well-known H-tree construction (see for example, page 84 of [12]) the complete binary tree with n leaves is embedded in the $(2\sqrt{n}-1)\times(2\sqrt{n}-1)$ mesh and the maximum root-to-leaf distance in the mesh image is $2\sqrt{n}-2$. Of course, this embedding was not designed to satisfy our criteria and would in any case be very costly in terms of unused processor sites. In the embedding of [6], although the complete binary tree with n leaves is embedded in the square mesh with n nodes, the maximum root to leaf distance is $3.54\sqrt{n}$. Moreover, only tree edges at the same depth are guaranteed to be mapped to disjoint paths.

Compared with other previous embeddings and for some P-RAM algorithms, the edge-disjointness property of our embeddings in the mesh yields further complexity gains. Occasionally it is useful for all nodes in the tree, not just those at the same level, to pass messages simultaneously to their children in such a way that this continues until all messages (including that from the root) reach the leaves of the tree. An example of such a *cascading* requirement is provided within the implementation of a bracket matching algorithm on a mesh detailed in [6]. This can be simulated in the embedding of [6] by allowing the messages from the internal tree nodes adjacent to leaves to be passed directly to the leaves, then subsequently messages from the nodes at the next level are sent to the leaves and so on, until finally the message from the root is allowed to be copied down to all descendants. In this way, only tree edges at the same level are being used at the same time and the lack of disjointness of all paths from the root to the leaves is no hindrance. In the embedding of [6], the routing time for such a process would be $3.54(1+1/2+1/4+1/8+\cdots)\approx 7.08\sqrt{n}$. The successive terms arise from routing from successive tree levels. For the embedding of this paper, the path-disjointness property allows messages to be passed down the tree simultaneously from all levels, and so the routing time for the *cascading* requirement is just that for passing a message from the root to the leaves (this masks the time for message passing from all other internal nodes) which is $\sqrt{n}$.

5 Summary and open problems

We have described dense edge-disjoint embeddings of the complete binary tree with n leaves in the following n-node intercommunication networks: the hypercube, the de Bruijn and shuffle-exchange graphs and the two-dimensional mesh. The embeddings have the following properties: paths of the tree are mapped onto edge-disjoint paths of the host graphs, and at most two tree nodes (just one of which is a leaf) are mapped onto each host node. We also proved (except for the shuffle-exchange graph) that an algorithmically important parameter, the maximum distance from a leaf to the root of the tree, is asymptotically as short as possible. We conjecture that for the shuffle-exchange graph this distance is also optimally short within our embedding. The embeddings facilitate efficient implementation of many P-RAM algorithms on these networks and improve extant results. For the mesh and shuffle-exchange graphs these embeddings required replacing each edge by a pair of parallel (or anti-parallel) edges.

A number of problems remain open. Because of the logarithmic lower order term in $P(n)$ for the embedding of a complete binary tree in the mesh there is a small gap between the distance obtained here and the naive lower bound of the network radius. Whether this gap can be closed, from either side, is an open question. A mesh architecture sometimes used is in the form of a torus, with no boundary. It seems unlikely that a complete binary tree with 2^{2m} leaves could be embedded in the directed $2^m \times 2^m$ torus, but we have not been able to prove this. For the shuffle-exchange graph, we conjecture that our embedding give the asymptotically shortest possible maximum root-to-leaf distance consistent with our embedding requirements.

The question of how to find similarly dense embeddings of complete binary trees in meshes of higher dimension is unanswered. Similarly, the question of finding dense embeddings of complete trees of fixed higher degree in various interconnection networks is unsolved. From a graph-theoretic point of view, the dense edge-disjoint embedding of *arbitrary* trees in networks presents a challenge, although these problems may prove to be of less general algorithmic importance than embedding complete trees.

References

[1] S. N. Bhatt and I. C. F. Ipsen, "How to Embed Trees in Hypercubes", *Report DCS/RR–443*, Dept. of Computer Science, Yale University (1985).

[2] A. M. Gibbons, "An introduction to distributed memory models of parallel computation", Chapter 10 of *Lectures on parallel computation*, editors: A. M. Gibbons and P. Spirakis, Cambridge University Press (1993).

[3] A. M. Gibbons and M. S. Paterson, "Dense edge-disjoint embedding of binary trees in the mesh", *4th Annual ACM Conference on Parallel Algorithms and Architectures (SPAA92)*, ACM Press (1992), 257–263.

[4] A. M. Gibbons and W. Rytter, *Efficient Parallel Algorithms*, Cambridge University Press (1988).

[5] A. M. Gibbons and Y. N. Srikant, "A class of problems efficiently solvable on the mesh-connected computer including dynamic expression evaluation", *Information Processing Letters* 37 (1991) 305–311.

[6] A. M. Gibbons and R. Ziani, "The balanced binary tree technique on mesh-connected computers", *Information Processing Letters*, 37 (1991), 101–109.

[7] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*, Morgan Kaufman Publishers, San Mateo, California (1992).

[8] B. Monien and H. Sudborough, "Comparing interconnection networks", *Lecture Notes in Computer Science 324*, Proceedings of Mathematical Foundations of Computer Science 1988 (editors: M. P. Chytil, L. Janiga, V. Koubek), Springer-Verlag 138–153 (1988).

[9] D. Nassimi and S. Sahni, "An optimal routing algorithm for mesh-connected parallel computers", *Journal of the ACM* 27 (1980), 6–29.

[10] S. Ravindran and A. M. Gibbons, "Dense edge-disjoint embedding of binary trees in the hypercube", *Information Processing Letters* 45 (1993), 321–325.

[11] D. B. Skillicorn, "Architecture independent computation", *IEEE Computer* 23 (1990) 38–50.

[12] J. D. Ullman, *Computational Aspects of VLSI* Computer Science Press, Inc. (1984).

[13] L. G. Valiant, "A Bridging Model for Parallel Computation", *Communications of the ACM* 33 (1990), 103–111.

[14] L. G. Valiant and G. J. Brebner, "Universal schemes for Parallel Computation", *Proceedings of the 13th ACM Symposium on Theory of Computing* ACM Press (1981), 263–277.