INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Optimal Dynamic Embeddings of Complete Binary Trees into Hypercubes

Volker Heun Ernst W. Mayr*

TR-98-022

August 1998

Abstract

The double-rooted complete binary tree is a complete binary tree where the root is replaced by an edge. It is folklore that the double-rooted complete binary tree is a spanning tree of the hypercube of the same size. Unfortunately, the usual construction of an embedding of a double-rooted complete binary tree into the hypercube does not provide any hint how this embedding can be extended if each leaf spawns two new leaves. In this paper, we present simple dynamic embeddings of double-rooted complete binary trees into hypercubes which do not suffer from this disadvantage. We also present edge-disjoint embeddings of large binary trees with optimal load and unit dilation. Furthermore, all these embeddings can be efficiently implemented on the hypercube itself such that the embedding of each new level of leaves can be computed in constant time. Since complete binary trees are similar to double-rooted complete binary trees, our results can be immediately transfered to complete binary trees.

*Institut für Informatik der TU München, D-80290 München, Germany

1 Introduction

Hypercubes are a very popular model for parallel computation because of their regularity and their relatively small number of interprocessor connections. Another important property of an interconnection network is its ability to efficiently simulate the communication of parallel algorithms. Thus, it is desirable to find suitable embeddings of graphs representing the communication structure of parallel algorithms into hypercubes representing the interconnection network of a parallel computer.

Most embeddings are constructed as *static* embeddings which means that the whole information about the structure of the guest graph is known in advance. Since the guest graph represents the communication structure of a parallel algorithm, the guest graph may vary during the execution of the algorithm. Thus, it is important to investigate *dynamic* embeddings of graphs.

Static embeddings are usually much easier to construct than dynamic embeddings. Moreover, it might be impossible to construct dynamic embeddings deterministically with high quality. For arbitrary binary trees, it has been proven that dynamic embeddings cannot be constructed with high quality if neither randomization nor migration, i.e., remapping of tree vertices, is allowed [7].

Often complete binary trees represent the communication structure of parallel algorithms, e.g., divide-and-conquer or branch-and-bound algorithms. Embeddings of complete binary trees have been investigated by numerous researchers. Unfortunately, the complete binary tree is not a subgraph of the hypercube of nearly the same size [1, 10]. But using the fact that the double-rooted complete binary tree is a spanning tree of the hypercube of the same size, optimal embeddings of complete binary trees into hypercubes have been discovered [1, 2, 3, 5, 8, 9, 11, 12, 13]. The major drawback of these algorithms is that they are constructed as static embeddings and that they do not provide any hints how these embeddings can be extended if the tree grows by a complete level. Only in an unpublished paper, dynamic embeddings of complete binary trees into hypercubes have been investigated in [4].

In this paper, we focus on dynamic embeddings of double-rooted complete binary trees into hypercubes implying dynamic embeddings of complete binary trees into hypercubes. Our construction improves and simplifies the results in [4]. We assume that at each step the double-rooted complete binary tree can grow or shrink by a complete level of its leaves. Therefore, the size of the double-rooted complete binary tree will be doubled or halved. If dynamic embeddings into optimal hypercubes are considered, then for each new level one half of the old leaves must be remapped to obtain a unit dilation embedding. This is optimal in the sense that we cannot achieve unit dilation and unit load, if less vertices are remapped.

We also consider embeddings of large binary trees. Here, it is important to obtain embeddings with small congestion, since occasionally it is necessary to pass messages along all tree edges. A first suboptimal approach is given in [3], where embeddings with optimal load but non-optimal congestion has been constructed. The first optimal embedding is given in [9] with respect to load, dilation, and congestion. Nevertheless, both algorithms suffer from their static design.

We present a dynamic edge-disjoint embedding such that the vertices of the same level are distributed evenly among the hypercube vertices and the load is optimal. Considering embeddings into d-dimensional hypercubes, it will be shown that a double-rooted complete binary tree of height at most $d+\lfloor\log(\frac{d+6}{5})\rfloor$ can be embedded with unit dilation, unit congestion, and optimal load, where siblings are mapped to different hypercube vertices. Moreover, we show that all these embeddings can be computed on the hypercube itself spending only constant time for each new level. Thus, we present for the first time a dynamic algorithm for optimal embedding complete binary trees into hypercubes of arbitrary size.

The remainder of this paper is organized as follows. In the next section, we will give the basic definitions and notations. A brief overview of the known results of embeddings of complete binary trees into hypercubes will be given in the third section. We then will show that the embedding of a double-rooted complete binary tree into its optimal hypercube can be constructed level by level spending only constant time for each new level. Embeddings with load greater than one and unit congestion are presented in the fifth section. Extensions to sibling-separating embeddings will be discussed in the sixth section. Finally, we give some concluding remarks.

2 Basic Definitions and Notations

An *embedding* of a graph $G{=}(V_G, E_G)$ into a graph $H{=}(V_H, E_H)$ is a pair of mappings $\varphi{:}V_G{\rightarrow}V_H$ and $\psi{:}E_G{\rightarrow}\mathcal{P}(H)$. Here, $\mathcal{P}(G)$ denotes the set of paths in the graph $G{=}(V, E)$. The mapping ψ maps each edge $\{v, w\}{\in}E_G$ to a path $p{\in}\mathcal{P}(H)$ connecting $\varphi(v)$ and $\varphi(w)$. The graph G is called *guest graph* and the graph H *host graph.* We call an embedding *one-to-one* if the mapping φ is 1-1. The ratio $|V_H|/|V_G|$ is called the *expansion* of the embedding. The *dilation* of an edge $e{\in}E_G$ is the length of the path $\psi(e)$. Here, the length of a path p is the number of its edges. The *dilation of an embedding* is the maximal dilation of an edge in G. The number of vertices of a guest graph which are mapped onto a vertex v in the host graph, is called the *load* of the vertex v. The *load of an embedding* is the maximal load of a vertex in the host graph. The *congestion* of an edge $e'{\in}E_H$ is the number of paths in $\{\psi(e) \mid e{\in}E_G\}$ that contain e'. The *congestion of an embedding* is the maximal congestion over all edges in H.

A *hypercube of dimension d* is a graph with 2^d vertices, labeled 1-1 with the strings in $\{0,1\}^d$. Two vertices are connected iff their labels differ in exactly one position. An edge belongs to dimension i, if the labels of the vertices incident to that edge differ in position i. Here, the bit positions are numbered from 1 to d in left-to-right order. Let v be a label of a hypercube vertex, we denote by v^i the label which differ in position i from v. The smallest hypercube into which a graph can be embedded with unit load is called its *optimal* hypercube. For a graph $G{=}(V,E)$, the dimension of the optimal hypercube is $\lceil \log(|V|) \rceil$. Thus, an embedding of a graph into its optimal hypercube has expansion less than two.

As usual a *tree* is a connected acyclic graph with a distinguished vertex, called the *root*. A vertex of degree 1 is called a *leaf* of the tree if it is not the root. A vertex is called an *internal* vertex if it is not a leaf. A vertex w is called a *child* of a vertex v if w is adjacent to v and w does not belong to the simple path from the root to v. The vertex v is also called a *parent* of w. The *level* of a vertex v in a tree is the number of vertices on the simple path from the root to v. Hence, the level of the root is 1. The *height* of a tree T is the maximum level of a vertex in T. In the following, we consider ordered trees, i.e., we distinguish between left and right vertices. A vertex is called a *left* (resp., *right*) vertex if it is the left (resp., right) child of its parent.

A *complete binary tree* T of height h is a tree such that each internal vertex has exactly two children, and such that all leaves have the same level. A *double-rooted*

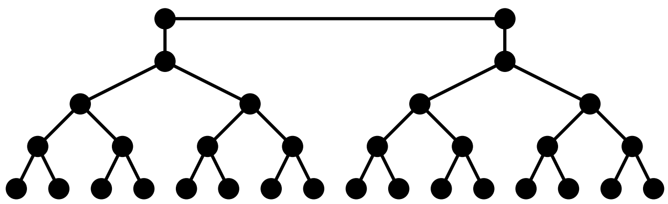

Figure 1: A Double-Rooted Complete Binary Tree of Height 5

complete binary tree of height h (or briefly a DRCBT) is a complete binary tree of height h where the root is replaced by a path of length 1 (cf. Figure 1). We will refer to both of these vertices as roots which both are assumed to have level 1. A double-rooted complete binary tree of height h can also be viewed as two complete binary trees of height $h{-}1$ whose roots are connected by a path of length 3. In the following, we will call this path the *root path*. Clearly, a double-rooted complete binary tree of height h has exactly 2^h vertices.

3 Previous Work

Unfortunately, the complete binary tree is not a subgraph of its optimal hypercube [1, 10]. We recall that both the complete binary tree and the hypercube are bipartite graphs. Given a bipartite graph $G{=}(A, B, E)$ of size n, we call G *balanced* if $|A|, |B| \leq 2^{\lceil \log(n) \rceil - 1}$. Obviously, the hypercube is a balanced bipartite graph, where the set of vertices can be partitioned as required into the sets of vertices whose labels have an odd or an even number of 1's, respectively.

Lemma 1 *A graph G is a subgraph of its optimal hypercube only if G is a balanced bipartite graph.*

Thus, only balanced trees can be embedded with a unit dilation and unit load into their optimal hypercubes. The complete binary tree of height greater than 2 is not balanced which can be seen as follows. Let $T{=}(A, B, E)$ be a complete binary tree of height $h{\geq}3$ and, therefore, of size $2^h{-}1$. Without loss of generality, we assume that the leaves of T are contained in A. Hence, we obtain $|A| \geq 2^{h-1}{+}2^{h-3} > 2^{h-1} = 2^{\lceil \log(2^h-1) \rceil - 1}$ implying that T is not balanced.

Theorem 2 ([5, 1, 10]) *A complete binary tree cannot be embedded into its optimal hypercube with unit dilation and unit load.*

An optimal embedding of complete binary trees into their optimal hypercubes was first discovered by Havel and Liebl [5] and was also exhibited by Bhatt and Ipsen [1] as well as by Saad and Schultz [10]. We will give here a short proof of this well known fact using the double-rooted complete binary tree.

In what follows, we will show that the double-rooted complete binary tree is a subgraph of its optimal hypercube. This will be proved by induction on the height of the double-rooted complete binary tree. Indeed, we proof a little bit stronger statement. Additionally, we require that the root path traverses hypercube edges of three different dimensions. For $h{=}3$ the claim is true, as can be seen from Figure 2.

Assume that the claim holds for a double-rooted complete binary tree of height h. An $h{+}1$-dimensional hypercube can be viewed as two copies of h-dimensional hypercubes. By induction hypothesis, each of the h-dimensional hypercubes contains a double-rooted complete binary tree of height h as a subgraph. We assume that the root path of the first double-rooted complete binary tree traverse hypercube edges in the following order of dimensions: i, j, and k. Without loss of generality, the root path of the second double-rooted complete binary tree traverse hypercube edges in the following order of dimensions: j, k, and i (cf. Figure 3(a)). Now, we can combine

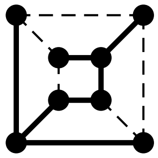

Figure 2: Embedding a DRCBT of Height 3 into Its Optimal Hypercube

these two double-rooted complete binary trees of height h to a new double-rooted complete binary tree of height $h+1$ as can be seen in Figure 3(b). Moreover, we can conclude from Figure 3 that the root path of the new double-rooted complete binary tree of height $h+1$ traverses hypercube edges of three different dimensions, namely j, $h+1$, and k.

Theorem 3 ([5, 1, 10]) *A double-rooted complete binary tree of height h can be embedded with unit dilation and unit load into its optimal hypercube.*

Since a double-rooted complete binary tree of height $h+1$ was constructed from a complete binary tree of height h by replacing the root with a path of length 1, we have also proved the following corollary which is optimal because of Theorem 2.

Corollary 4 *A complete binary tree of height h can be embedded into its optimal hypercube with unit load and dilation 2, where all but one edges have unit dilation.*

As mentioned earlier, a double-rooted complete binary tree of height $h+1$ consists of two complete binary trees of height h whose roots are connected by a path of length

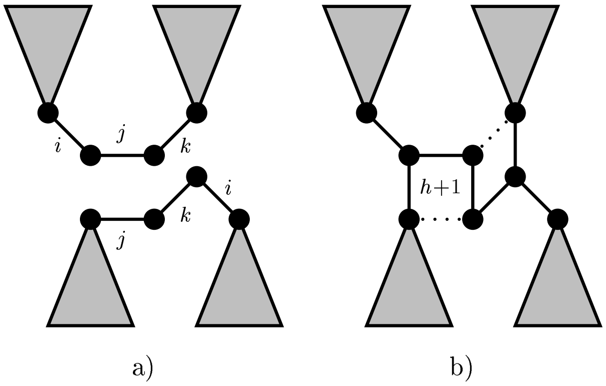

Figure 3: Embedding a DRCBT into Its Optimal Hypercube

3. Thus, Theorem 3 implies the following corollaries which are also optimal because of Theorem 2.

Corollary 5 *A complete binary tree of height h can be embedded with unit dilation and unit load into an $h+1$-dimensional hypercube.*

Corollary 6 *A complete binary tree of height h can be embedded with unit dilation and load 2 into its optimal hypercube.*

4 Optimal Dynamic One-To-One Embeddings

In this section, we present two simple level by level algorithms which construct the embedding of a double-rooted complete binary tree into its optimal hypercube given in the previous section. The first algorithm embeds the double-rooted complete binary tree level by level into its optimal hypercube, i.e., the first ℓ levels are embedded in an ℓ-dimensional subcube. The major drawback of this algorithm is that for each new level some leaves must be remapped. This will be avoided in the second algorithm. But in each step we then need a hypercube which is twice as large as necessary except when the final level is reached, where we get an embedding into its optimal hypercube. In this section, we consider one-to-one embeddings.

Unfortunately, the construction of the embedding mentioned in the previous section combines two embeddings of two double-rooted complete binary trees to a new embedding of a double-rooted complete binary tree of increased height. As mentioned earlier, we assume that the trees grow at the leaves and not at the root. We now present a simple algorithm which extends an embedding of a double-rooted complete binary tree of height h into an embedding of double-rooted complete binary tree of height $h+1$ such that only one half of the leaves have to be remapped. This is optimal which can be seen as follows. As the tree grows by one level, each leaf spawns two new leaves, but the dimension of the hypercube increases only by one. Hence, it is impossible to embed these new leaves with unit dilation without a remapping of any internal vertex. Since each leaf spans two new leaves, at least one quarter of tree vertices must be remapped to obtain an embedding with unit dilation.

Theorem 7 *The double-rooted complete binary tree of height h can be dynamically embedded into its optimal hypercube with unit dilation, unit load, and unit congestion such that for each new level of leaves only one half of the old leaves have to be remapped. The computation time for each new level is constant.*

Corollary 8 *The complete binary tree of height h can be dynamically embedded into its optimal hypercube with dilation 2, unit load, and unit congestion such that for each new level of leaves only one half of the old leaves have to be remapped. Moreover, all but one tree edges have unit dilation. The computation time for each new level is constant.*

Note that these theorems are optimal. We will give here a simple construction and a proof of this embedding. To prove the theorem, we will prove a stronger claim on the embedding.

A one-to-one embedding of the double-rooted complete binary tree of height h into its optimal hypercube with unit dilation can be constructed such that there exists a partition of the vertices of the double-rooted complete binary tree into 2^{h-2} groups of four vertices each fulfilling the following conditions:

i) exactly two of the four vertices are leaves of the double-rooted complete binary tree, more precisely, a left and a right leaf;

ii) one of the other two internal vertices is the parent of the right leaf;

iii) the four vertices are mapped to a 2-dimensional subcube, where the leaf and its parent fulfilling condition ii) are mapped to adjacent hypercube vertices as well as both leaves.

Obviously, the double-rooted complete binary tree of height 2 can be embedded into the 2-dimensional hypercube satisfying the three conditions above. This can also be seen from the left part of Figure 4, where the white vertices are the leaves and the gray and black vertex are the roots. As induction hypothesis, the black vertex in Figure 4

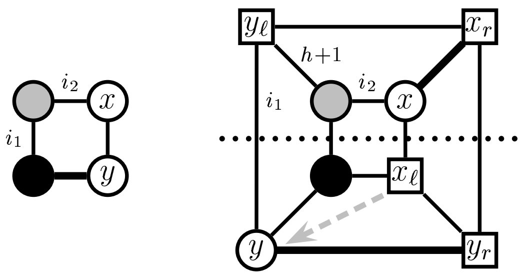

Figure 4: A DRCBT Spawning New Leaves in Its Optimal Hypercube

is the parent of its leaf as required in condition *ii)*; the gray vertex is another internal vertex of the double-rooted complete binary tree.

To obtain the embedding of the double-rooted complete binary tree of height $h{+}1$ into the $h{+}1$-dimensional hypercube the old mapping is extended as follows. As mentioned earlier, we have to remap one half of the leaves. All other vertices of the double-rooted complete binary tree of height h keep their hypercube vertices in the hypercube. Since we know that one leaf is adjacent in the hypercube to its parent, we remap this leaf such that it is again adjacent to its parent in the hypercube along the new dimension $h{+}1$ (cf. Figure 4). Now each leaf can spawn two new children such that they are adjacent to their parents in the optimal hypercube as can be seen from Figure 4, where the newly spawned leaves are drawn as squares. It can easily be verified that the required three conditions are satisfied by splitting the 3-dimensional subcube into an upper and lower 2-dimensional subcube as indicated in Figure 4.

From the proof of the embedding we get the following construction. Each leaf v stores the following *subcube information* $\sigma(v){=}(i_1, i_2)$, where i_1 and i_2 are the hypercube dimensions according to condition *iii)* and i_1 is the hypercube dimension connecting the two leaves. Note that the two leaves contained in the same subcube according to condition *iii)* have the same subcube information. In what follows, we denote by $\mathrm{loc}_h(v)$ the label of the hypercube vertex v in an h-dimensional subcube.

First, we consider as induction basis a double-rooted complete binary tree of height 2. Let r_1 and r_2 be the roots of the double-rooted complete binary tree and let x and y be their children, respectively. Then we get the following mapping for a double-rooted complete binary tree of height 2:

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\mathrm{loc}_2(r_1) & = & 00 & & & \\
\mathrm{loc}_2(r_2) & = & 10 & & & \\
\mathrm{loc}_2(x) & = & 01 & \sigma(x) & = & (1,2) \\
\mathrm{loc}_2(y) & = & 11 & \sigma(y) & = & (1,2)
\end{array}$$

The labels of the hypercube vertices to which the new leaves of the double-rooted complete binary tree of height $h{+}1$ are mapped can be computed from the labels of the hypercube vertices to which the vertices of the double-rooted complete binary tree of height h are mapped in constant time as follows. We assume that the subcube information σ for the vertices x, y are $\sigma(x){=}\sigma(y){=}(i_1, i_2)$. Note that x is a left leaf and y is a right leaf. Let x_ℓ, x_r, y_ℓ, and y_r be the newly spawned left and right vertices of x and y, respectively.

$$\begin{array}{rcr@{\qquad\qquad}rcl}
\mathrm{loc}_{h+1}(v) & = & \mathrm{loc}_h(v)\cdot 0 & \multicolumn{3}{l}{\text{for all internal vertices } v} \\[6pt]
\mathrm{loc}_{h+1}(x) & = & \mathrm{loc}_h(x)\cdot 0 & & & \\
\mathrm{loc}_{h+1}(x_\ell) & = & (\mathrm{loc}_h(x))^{i_2}\cdot 0 & \sigma(x_\ell) & = & (i_1, h+1) \\
\mathrm{loc}_{h+1}(x_r) & = & \mathrm{loc}_h(x)\cdot 1 & \sigma(x_r) & = & (h+1, i_1) \\
\mathrm{loc}_{h+1}(y) & = & (\mathrm{loc}_h(y))^{i_1}\cdot 1 & & & \\
\mathrm{loc}_{h+1}(y_\ell) & = & ((\mathrm{loc}_h(y))^{i_1})^{i_2}\cdot 1 & \sigma(y_\ell) & = & (h+1, i_1) \\
\mathrm{loc}_{h+1}(y_r) & = & \mathrm{loc}_h(y)\cdot 1 & \sigma(y_r) & = & (i_1, h+1)
\end{array}$$

If the height of the double-rooted complete binary tree is known in advance or if we can determine when the final level is reached, we can clearly compute the final hypercube vertex of each newly spawned leaf. We will denote by h the maximal level, i.e., the height of the double-rooted complete binary tree. Thus, in this case a remapping is not necessary although in the intermediate steps a hypercube of one dimension larger is used than actually needed. This extension of the mapping is

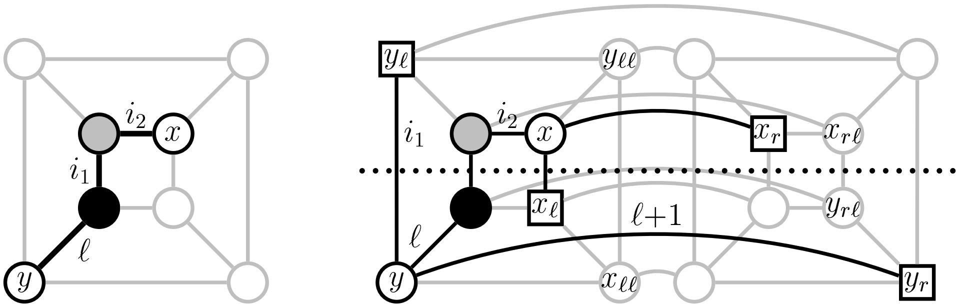

Figure 5: A DRCBT Spawning New Leaves at Level ℓ in a Larger Hypercube

illustrated in Figure 5. Actually, also the embedding of the left children of the new leaves are illustrated in this figure which will be useful in a later discussion. From this illustration, the following construction can immediately be obtained. Note that always $\ell{-}1{\in}(i_1, i_2){=}\sigma(x){=}\sigma(y)$ holds.

$$\begin{array}{rclcrcl}
\mathrm{loc}_h(x_\ell) & = & (\mathrm{loc}_h(x))^{i_2} & \qquad & \sigma(x_\ell) & = & (i_1, \ell) \\
\mathrm{loc}_h(x_r) & = & (\mathrm{loc}_h(x))^{\min\{\ell+1,h\}} & & \sigma(x_r) & = & (\ell, i_1) \\
\mathrm{loc}_h(y_\ell) & = & (\mathrm{loc}_h(y))^{i_2} & & \sigma(y_\ell) & = & (\ell, i_1) \\
\mathrm{loc}_h(y_r) & = & (\mathrm{loc}_h(y))^{\min\{\ell+1,h\}} & & \sigma(y_r) & = & (i_1, \ell)
\end{array}$$

Theorem 9 *The double-rooted complete binary tree of height h can be embedded dynamically into its optimal hypercube with unit dilation, unit load, and unit congestion without remapping of any vertex. The computation time for each new level is constant.*

Corollary 10 *The complete binary tree of height h can be embedded dynamically into its optimal hypercube with dilation 2, unit load, and unit congestion without remapping of any vertex. Moreover, all but one tree edges have unit dilation. The computation time for each new level is constant.*

5 Optimal Dynamic Edge-Disjoint Embeddings

If we consider level by level computations on double-rooted complete binary trees or on complete binary trees, it might be useful to construct embeddings such that

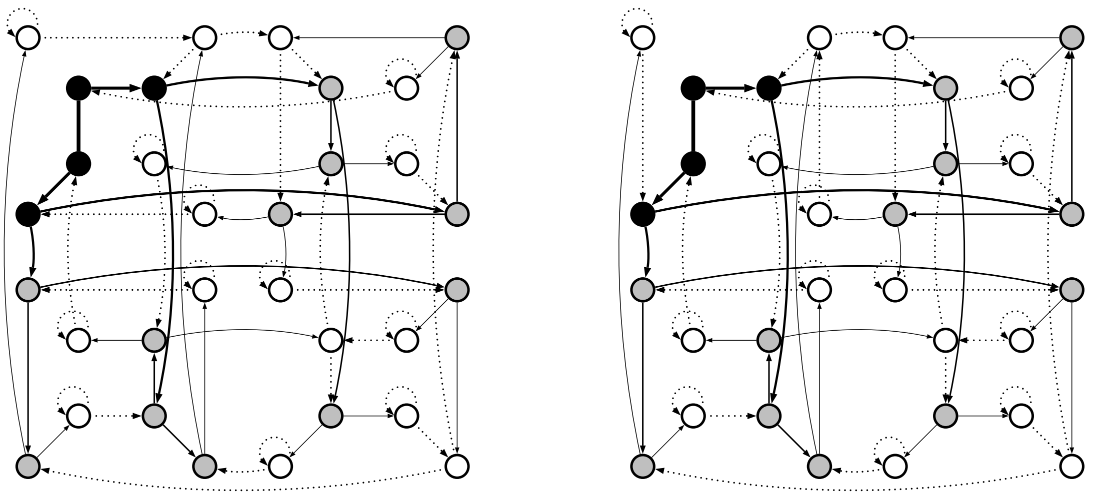

Figure 6: Two Embeddings of a DRCBT of Height 6 into a 5-dimensional Hypercube [9]

only vertices of any fixed level are mapped to different hypercube vertices. Thus, we are interested in a load 2 embedding such that adjacent tree vertices are mapped to adjacent hypercube vertices or to the same hypercube vertex.

A first simple approach to obtain a dynamic embedding of the double-rooted complete binary tree of height $h+1$ into an h-dimensional hypercube with unit dilation and load 2 is to embed the double-rooted complete binary tree of height h into an h-dimensional hypercube dynamically. If a leaf v at level h spawns itself two new leaves v_ℓ and v_r, we map v_ℓ to the hypercube vertex labeled $\mathrm{loc}_h(v)$ and v_r to the hypercube vertex labeled $(\mathrm{loc}_h(v))^{i_2}$, where $\sigma(v)=(i_1, i_2)$.

Theorem 11 *A double-rooted complete binary tree of height $h+1$ can be dynamically embedded into an h-dimensional hypercube with unit dilation, load 2 and congestion 2. Vertices of the same level are mapped to different hypercube vertices. The computation time for each new level is constant.*

With respect to the congestion, we can do even better. An embedding of a double-rooted complete binary tree of height 6 into a 5-dimensional hypercube with unit dilation, unit congestion, and load 2 was discovered by Ravindran and Gibbons [9], which is illustrated in Figure 6. Actually, Figure 6 shows two different but quite similar embeddings, where the second one will be needed later. The root path is indicated by black nodes while the parents of the leaves are drawn white. Note that the root path traverses three different dimensions as our four vertices in the induction step of our embedding in the previous section. For convenience, the edges incident to the leaves are drawn as dotted arcs. Using this construction, we can find a dynamic embedding for a double-rooted complete binary tree of height at least 6 into a hypercube with unit congestion. Thus, we obtain an embedding with unit dilation, unit congestion and load 2, if we modify our embedding in the last four levels as indicated in Figure 6. Obviously, the embedding of these lower levels can be computed in constant time for each new level.

Theorem 12 *A double-rooted complete binary tree of height $h+1 \geq 6$ can be dynamically embedded into an h-dimensional hypercube with unit dilation, unit congestion, and load 2. Vertices of the same level are mapped to different hypercube vertices. The computation time for each new level is constant.*

Clearly, this theorem is optimal. As mentioned in the paper of Ravindran and Gibbons [9], the last theorem do not hold for trees of height less than 6. The theorem above can be easily extended to embeddings with higher load as follows. Note that the embedding of a double-rooted complete binary tree of height $h+1$ into an h-dimensional hypercube has congestion 1 and every hypercube vertex is an image of exactly one leaf. Hence, these embedding can be extended for arbitrary load, if we map the children of high levels to same hypercube vertex as their parents.

Theorem 13 *A double-rooted complete binary tree of height $h \geq 6$ can be dynamically embedded into a d-dimensional hypercube with unit dilation, unit congestion, and load 2^{h-d}. Vertices of a fixed level $\ell \leq d+1$ are mapped to different hypercube vertices; vertices of a fixed level $\ell \geq d+1$ are distributed evenly among all hypercube vertices. The computation time for each new level is constant.*

Corollary 14 *A complete binary tree of height $h \geq 6$ can be dynamically embedded into a d-dimensional hypercube with dilation 2, unit congestion, and load 2^{h-d}. Vertices of a fixed level $\ell \leq d+1$ are mapped to different hypercube vertices; vertices of a fixed level $\ell \geq d+1$ are distributed evenly among all hypercube vertices. Moreover, all but one tree edges have unit dilation. The computation time for each new level is constant.*

6 Dynamic Edge-Disjoint Sibling-Separating Embeddings

The embedding stated in Theorem 13 can be extended such that siblings are mapped to different hypercube vertices maintaining unit congestion. Furthermore, tree vertices from a subtree rooted at level d are mapped to different hypercube nodes. Therefore, this embedding allows efficient simulations of tree-algorithms on more irregularly structured binary trees than before.

In what follows, we will show that each 10-dimensional subcube formed by consecutive dimensions contains a spanning 2-regular subgraph which avoids already used edges. Such a spanning 2-regular subgraph allows the following embedding strategy. For an already mapped tree vertex, we map its left child to the same hypercube node and its right child to the hypercube node given by the successor in the spanning 2-regular subgraph interpreted as a set of directed cycles. First we state two basic technical facts which we will use later. As usual, a *matching* in a graph is a subset of edges such that each vertex in the graph is incident to at most one matched edge. A matching is called *perfect* if each graph vertex is incident to a matched edge.

Lemma 15 *Let $H{=}(V,E)$ be a 3-dimensional hypercube and $M{\subset}E$ a matching. Then $H'{=}(V,E\backslash M)$ contains a perfect matching.*

Proof: If M is a perfect matching then H' is a 2-regular bipartite graph. Due to Hall's Theorem its edge set can be partitioned into two prefect matchings. Otherwise $|M|{\leq}3$ and there exists a dimension $i{\in}[1{:}3]$ such that M contains at most one edge of dimension i. Let M' be the set of edges of dimension i in H. If $M{\cap}M'{=}\emptyset$, we are done. Otherwise consider a 2-dimensional subcube (v_1,v_2,v_3,v_4) of H containing the edge $\{v_1,v_2\}{\in}M{\cap}M'$. Clearly, $M'\backslash\{\{v_1,v_2\},\{v_3,v_4\}\}{\cup}\{\{v_2,v_3\},\{v_4,v_1\}\}$ is a perfect Matching in H'. ■

Lemma 16 *Let $H{=}(V,E)$ be a 4-dimensional hypercube and $M{\subset}E$ a matching. Then $H'{=}(V,E\backslash M)$ contains a spanning 2-regular subgraph.*

Proof: If M is a perfect matching then H' is a 3-regular bipartite graph. Due to Hall's Theorem its edge set can be partitioned into three prefect matchings. Clearly, two of these perfect matchings form a 2-regular subgraph of H'. Otherwise, $|M|{<}8$ and without loss of generality we assume that M contains at most one edge of dimension 4. Now we split H into two 3-dimensional subcubes H_1 and H_2 by removing all edges of dimension 4. Let E_4 denote the set of edges belonging to dimension 4. Because of Lemma 15 H_1 and H_2 contain perfect matchings M_1 and M_2, respectively, such that M and $M'{:=}M_1{\cup}M_2$ are disjoint. Clearly, $M'{\cup}E_4$ form a 2-regular subgraph of H. If $M{\cap}E_4{=}\emptyset$, we are done. Otherwise, by the previous assumption $M{\cap}E_4{=}\{m\}$ for some edge $(v,v^4){=}m{\in}E_4$. Since v and v^4 have degree 4, there exists a dimension i such that $(v,v^4,v^{i,4},v^i)$ forms a cycle in H and $(v,v^i),(v^4,v^{i,4}){\notin}M'$. Now it can be easily seen that $C{=}M'{\cup}(E_4\backslash\{(v,v^4),(v^i,v^{i,4})\}){\cup}\{(v,v^i)(v^4,v^{i,4})\}$ is a spanning 2-regular subgraph such that C and M are disjoint. ■

Since the construction of the highest numbered four levels of the tree as shown in Figure 6 is somehow irregular compared with the embedding of the lower numbered levels, we will not reuse edges of dimension greater than $d{-}4$. In the sequel, we denote by $H_k(x,y)$ the k-dimensional subcube given by $\{z{\in}\{0,1\}^d \mid \exists w{\in}\{0,1\}^k{:}xwy{=}z\}$. We will call $H_k(\varepsilon,y)$ a *primary k-dimensional subcube* if $y{\in}10^*$.

First, we consider an $H_{10}(x,y)$ which is contained in an $H_{d-2}(\varepsilon,z)$ for $z{\neq}0^2$. Note that in Figure 6 all 3-dimensional subcubes except the upper left ones belong to such an $H_{d-2}(\varepsilon,z)$. Our embedding maps to these three subcubes given in Figure 6 tree vertices from level $d{-}1$, d, and $d{+}1$ using dimensions at least $d{-}3$ and one low dimension $i{<}d{-}3$. Thus, for each $u{\in}\{0,1\}^6$ the already used edges in an $H_4(x,uy)$ form a matching. By Lemma 16, we can conclude that each $H_4(x,uy)$ contains a spanning 2-regular subgraph and, hence, also each $H_{10}(x,y)$ contained in an $H_{d-2}(\varepsilon,z)$.

It remains to show that each $H_{10}(x,y)$ contained in $H_{d-2}(\varepsilon,0^2)$ contains a spanning 2-regular subgraph. By construction the dimensions used in the embedding given in Figure 6 are greater than $d{-}4$ with the exception of one reused low dimension $i{\leq}d{-}4$. This low dimension i is drawn in Figure 6 either as a straight horizontal or straight vertical line. By choosing the appropriate embedding given in Figure 6, we can ensure that the low dimension in the upper left 3-dimensional subcubes are not used. Thus, it is sufficient to consider only the embedding of the double-rooted complete binary tree up to level $d{-}3$. Now we are ready to compile some basic facts of the embedding stated in Theorem 12 up to level $d{-}3$. The following two lemmas follow immediately from a closer inspection of Figure 5

Lemma 17 *For $k{\leq}d{-}3$, a primary k-dimensional subcube gets tree vertices from levels k, $k{+}1$, and $k{+}2$.*

Lemma 18 *The set of dimensions across the neighbors of a tree vertex at level ℓ are mapped is either $\{i,\ell{-}1,\ell{+}2\}$, $\{i,\ell,\ell{+}2\}$, $\{i,\ell{+}1,\ell{+}2\}$, or $\{\ell,\ell{+}1,\ell{+}2\}$, for some $i{<}\ell{-}1$.*

Lemma 19 *Each $S{=}H_4(x,y)$ contains a spanning 2-regular subgraph if S is contained in a primary k-dimensional subcube for $k{\geq}|x|{+}6$.*

Proof: Since $H_4(x,y)$ is contained in a primary k-dimensional subcube, it follows from Lemma 17 that such an $H_4(x,y)$ gets tree vertices form level at least $|x|{+}6$. By Lemma 18, each node in $H_4(x,y)$ is incident to at most one edge which is an image of a tree edge. Thus, the already used edges form a matching and the claim follows immediately from Lemma 16. ■

Lemma 19 immediately implies that also each $H_{10}(x,y)$ contained in a primary subcube has a spanning 2-regular subgraph. It remains to show that for each $y{\in}0^*$

$H_{10}(x,y)$ contains a spanning 2-regular subgraph. For $u \in \{0,1\}^6 \setminus \{0,1\}^2 0^4$, again Lemma 19 implies that $H_4(x,uy)$ contains a spanning 2-regular subgraph. In the following, let $S_{00}{=}H_4(x,u00y)$, $S_{01}{=}H_4(x,u01y)$, and $S_{10}{=}H_4(x,u10y)$ for $u \in (0{+}1)^2 0^2$. In what follows, we will show that $S_{00} \cup S_{01} \cup S_{10}$ contains a spanning 2-regular subgraph. By Lemma 17, our embeddings maps to S_{00} only tree vertices from level at most $|x|{+}7$ while S_{01} and S_{10} gets tree vertices from level at least $|x|{+}9$. Hence, the edges between S_{00} and $S_{01} \cup S_{10}$ are not images of tree edges. Lemma 18 again implies that the used edges in S_{01} and S_{10} form a matching and thus by Lemma 15 we can find a perfect matching M_{01} and M_{10} of unused edges in S_{01} and S_{10}, respectively. Therefore, $M_{01} \cup M_{10}$ together with the edges between S_{00}, S_{01}, and S_{10} form a spanning 2-regular subgraph in $S_{00} \cup S_{01} \cup S_{10}$. Altogether we have proved that each $H_{10}(x,y)$ contains a spanning 2-regular subgraph avoiding already used edges.

It remains to compute the height of the double-rooted complete binary tree which can be embedded in this manner. The vertices of the double-rooted complete binary tree at level $d{+}1{+}\ell$ are distributed evenly among the hypercube vertices, i.e., each hypercube vertex is an image of exactly 2^ℓ tree vertices at level $d{+}1{+}\ell$. Since the required number of dimension is 10 times the load, we need $10{\cdot}2^\ell$ different dimensions for each new embedded level. Thus, the height of the tree which can be embedded in this manner is bounded by $d{+}2{+}k$, where k is the maximal value satisfying $\sum_{\ell=0}^{k} 10{\cdot}2^\ell \leq d{-}4$. This implies that $k{=}\lfloor \log(\frac{d+6}{20}) \rfloor$ and that the height of the double-rooted complete binary tree is bounded by $d{+}\lfloor \log(\frac{d+6}{5}) \rfloor$.

Theorem 20 *For $d{\geq}5$, a double-rooted complete binary tree of height $d{+}\lfloor \log(\frac{d+6}{5}) \rfloor$ can be dynamically embedded into a d-dimensional hypercube with unit dilation, unit congestion, and optimal load. Vertices of a fixed level $\ell{\leq}d{+}1$ are mapped to different hypercube vertices; vertices of a fixed level $\ell{\geq}d{+}1$ are distributed evenly among all hypercube vertices. Also each pair of siblings is mapped to two different hypercube vertices. The computation time for each new level is constant.*

Corollary 21 *For $d{\geq}5$, a complete binary tree of height $d{+}\lfloor \log(\frac{d+6}{5}) \rfloor$ can be dynamically embedded into a d-dimensional hypercube with dilation 2, unit congestion, and optimal load. Vertices of a fixed level $\ell{\leq}d{+}1$ are mapped to different hypercube vertices; vertices of a fixed level $\ell{\geq}d{+}1$ are distributed evenly among all hypercube vertices. Also each pair of siblings is mapped to two different hypercube vertices. The computation time for each new level is constant.*

A more detailed and sophisticated analysis shows that each 8-dimensional subcube formed by consecutive dimensions contains a spanning 2-regular subgraph which avoids already used edges. It can also been shown that six consecutive dimension are necessary for this approach.

7 Conclusion

We have presented some simple algorithms for dynamic embeddings of complete binary trees into hypercubes with optimal dilation, load, expansion, and congestion. The computations needed to embed a new level of leaves are very simple and can be computed in constant time. Our embeddings improve the known embeddings of complete binary trees into hypercubes.

It remains to be investigated whether arbitrary binary trees can be embedded dynamically into hypercubes. As mentioned earlier, we can expect dynamic embeddings with high quality only if we allow randomization or migration [7]. Allowing randomization, a dynamic embedding of arbitrary binary trees into their optimal hypercubes with dilation 8, and with high probability constant load and constant congestion are constructed [7]. On the other hand, we have found a dynamic embedding of arbitrary binary trees into optimal hypercubes with unit load, constant dilation and constant congestion if migration of vertices is allowed [6].

References

[1] S. Bhatt, I. Ipsen: How to Embed Tress in Hypercubes, Technical Report DCS/RR-443, Dept. of Computer Science, Yale University, 1985.

[2] Th. Bier, K.F. Loe: Embedding of Binary Trees into Hypercubes, *J Parallel Distrib. Comput.* **6** (1989) 679–691.

[3] K. Efe: Embedding Large Complete Binary Trees in Hypercubes with Load Balancing, *J. Parallel Distrib. Comput.*, **35** (1996) 104–109.

[4] T. Feder, E.W. Mayr: An Efficient Algorithm for Embedding Complete Binary Trees in the Hypercube, Manuscript, Dept. of Computer Science, Stanford University, 1987.

[5] I. Havel, P. Liebl: Embedding the Polytomic Tree into the n-Cube, *Časopis. Pěst. Mat.* **98** (1973) 307–314.

[6] V. Heun, E.W. Mayr: Efficient Dynamic Embeddings of Arbitrary Binary Trees into Hypercubes, in *Proc. 3rd Int. Workshop on Parallel Algorithms for Irregularly Structured Problems*, eds. A. Ferreira et al. (LNCS 1117, Springer, 1996), 287-298.

[7] T. Leighton, M. Newman, A. Ranade, W. Schwabe: Dynamic Tree Embeddings in Butterflies and Hypercubes, *SIAM J. Comput.* **21** (1992) 639–654.

[8] E. Leiss, H. Reddy: Embedding Complete Binary Trees into Hypercubes, *Inf. Process. Lett.* **38** (1991) 197–199.

[9] S. Ravindran, A. Gibbons: Dense Edge-Disjoint Embedding of Complete Binary Trees in the Hypercube, *Inf. Process. Lett.* **45** (1993) 321–325.

[10] Y. Saad, M. Schultz: Topological Properties of the Hypercube, Research Report RR-389, Dept. of Computer Science, Yale University, 1985.

[11] W.-K. Chen, M. Stallmann: On Embedding Binary Trees into Hypercubes, *J. Parallel Distrib. Comput.* **24** (1995) 132–138.

[12] A. Wagner: Embedding the Complete Tree in the Hypercube, *J. Parallel Distrib. Comput.* **20** (1994) 241–247.

[13] A. Wu: Embedding of Tree Networks into Hypercubes, *J. Parallel Distrib. Comput.* **2** (1985) 238–249.