TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

Volker Heun, Ernst W. Mayr

TUM-I9607
SFB-Bericht Nr.342/03/96 A
Januar 1996

TUM–INFO–01-96-I07-350/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

Volker Heun Ernst W. Mayr
Institut für Informatik
Technische Universität München
D-80290 München, Germany
{heun|mayr}@informatik.tu-muenchen.de
http://wwwmayr.informatik.tu-muenchen.de/

January 24, 1996

Abstract

The double-rooted complete binary tree is a complete binary tree where the path (of length 2) between the children of the root is replaced by a path of length 3. 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 and, therefore, of complete binary trees into hypercubes which do not suffer from this disadvantage. We also give an edge-disjoint embedding with optimal load 2 and unit dilation such that each hypercube vertex is an image of at most one vertex of a level. Moreover, 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, we can transfer our results immediately to complete binary trees. □

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 proved that dynamic embeddings cannot be constructed with high quality if neither randomization nor migration, i.e., remapping of tree vertices, is allowed [LNRS92].

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 many researchers. Unfortunately, the complete binary tree is not a subgraph of the hypercube of nearly the same size [SS85]. But using the fact that the double-rooted complete binary tree is a spanning tree of the hypercube of the same size, nearly optimal embeddings of complete binary trees into hypercubes have been discovered [HL73, BI85, LR91, RG93, W94]. The major drawback of these algorithms is that these are static embeddings. Only in [FM87] dynamic embeddings of complete binary trees into hypercubes have been investigated.

In this paper, we will focus on dynamic embeddings of double-rooted complete binary trees into hypercubes implying dynamic embeddings of complete binary trees into hypercubes which improves the results in [FM87]. 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 the optimal hypercube 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 with higher load. Here, it is important to obtain embeddings with small congestion, since occasionally it is necessary to pass messages along all tree edges. We will present an 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 h-dimensional hypercubes, we will show that we can embed a double-rooted complete binary tree of height at most $2h-1+\lfloor\log(h)\rfloor$ with unit dilation, optimal load, and optimal congestion 2, where siblings are mapped to different hypercube locations. Moreover, we will show that all these embeddings can be computed on the hypercube itself spending only constant time for every new level.

The remainder of this paper is organized as follows. In the next section, we will give the basic definitions and notations. A short 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 congestion at most 2 are presented in the fifth section. Finally, we give some concluding remarks.

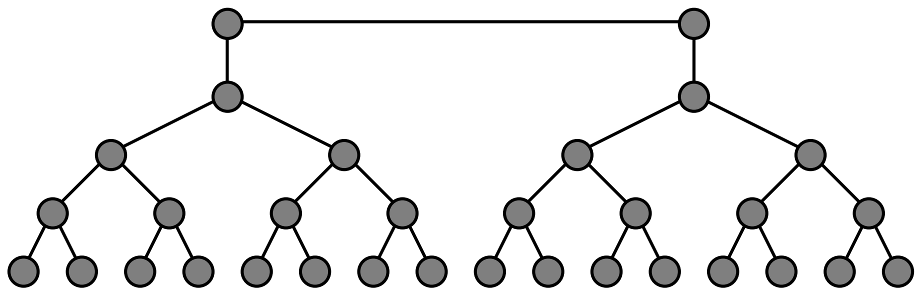

Figure 1: A Double-Rooted Complete Binary Tree of Height 5

2 Preliminaries

An *embedding* of a graph $G{=}(V_G, E_G)$ into a graph $H{=}(V_H, E_H)$ is a mapping $\phi{:}G{\to}H$ consisting of two mappings $\phi_V{:}V_G{\to}V_H$ and $\phi_E{:}E_G{\to}\mathcal{P}(H)$. Here, $\mathcal{P}(G)$ denotes the set of paths in the graph $G{=}(V, E)$. The mapping ϕ_E maps each edge $\{v, w\}{\in}E_G$ to a path $p{\in}\mathcal{P}(H)$ connecting $\phi_V(v)$ and $\phi_V(w)$. The graph G is called *guest graph* and the graph H *host graph*. We call an embedding *one-to-one* if the mapping ϕ_V is 1-1.

The *dilation* of an edge $e{\in}E_G$ under an embedding ϕ is the length of the path $\phi_E(e)$. Here, the length of a path p is the number of its edges. The *dilation of an embedding* ϕ 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* ϕ is the maximal load of a vertex in the host graph. The ratio $\frac{|V_H|}{|V_G|}$ is called the *expansion* of the embedding ϕ. The *congestion* of an edge $e'{\in}E_H$ is the number of paths in $\{\phi_E(e) \mid e{\in}E_G\}$ 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. Let v be a label of a hypercube location, we denote by v^i the label which differ in position i from the label v.

The smallest hypercube into which we can embed a graph $G{=}(V, E)$ with load one is called its *optimal* hypercube. The dimension of the optimal hypercube is $\lceil \log(|V|) \rceil$. Thus, an embedding of a graph G into its optimal hypercube has expansion less than two.

The *level* of a vertex v in a tree is the number of vertices on the 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.

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 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). 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*. Hence, a double-rooted complete binary tree of height h has exactly 2^h vertices.

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.

3 Previous Work

It would be nice if the complete binary tree is a subgraph of its optimal hypercube. Unfortunately, the complete binary tree is not a subgraph of its optimal hypercube [SS85]. 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. Hence, we have proved the following lemma.

Lemma 1 *A graph $G{=}(V, E)$ of size n is a subgraph of the $\lceil \log(n) \rceil$-dimensional hypercube only if G is a balanced bipartite graph.*

Thus, only balanced trees can be embedded with 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 *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 in [HL73] and was again exhibited in [BI85]. We will give here a short well known proof of this 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 traverse 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. A $h{+}1$-dimensional hypercube can be viewed as two copies of h-dimensional hypercubes. By induction hypothesis, each of the h-dimensional hypercubes contain a double-rooted complete binary tree of height h as a subgraph. We assume that the root path of the first

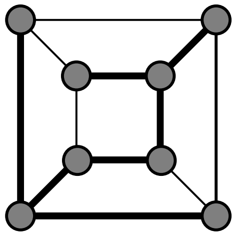

Figure 2: Embedding a DRCBT of Height 3 into Its Optimal Hypercube

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 these two double-rooted complete binary trees of height h to a new double-rooted complete binary tree of height $h+1$ as shown 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 *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. Moreover, all but one edges have unit dilation.*

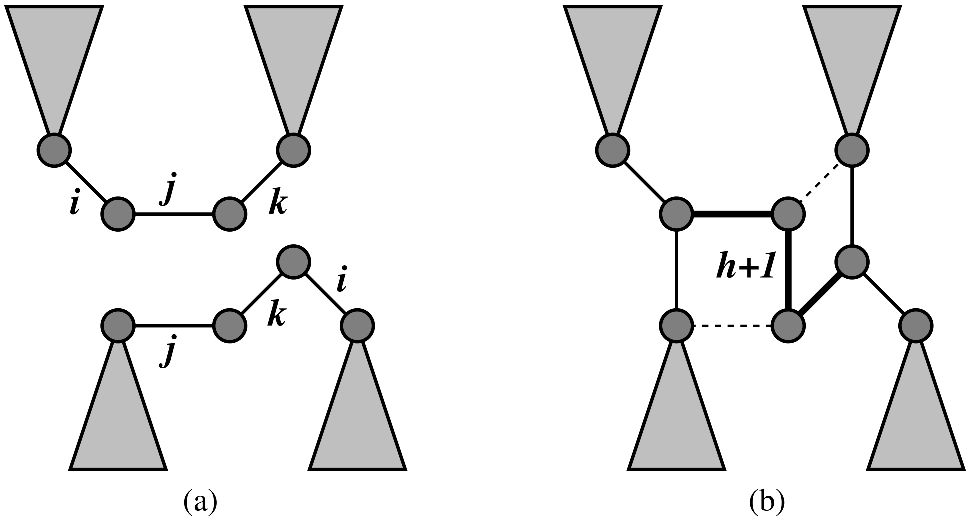

Figure 3: Inductive Step in the Embedding of a DRCBT into Its Optimal Hypercube

As mentioned earlier, each 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 3. Again, 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 a $h+1$-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 Dynamic Embeddings with Unit Load

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 have been embedded in a 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 double 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, unless noted otherwise.

Unfortunately, the construction of the embedding 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 grows 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 spans 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, it is clear that at least a quarter of tree vertices must be remapped to obtain 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*

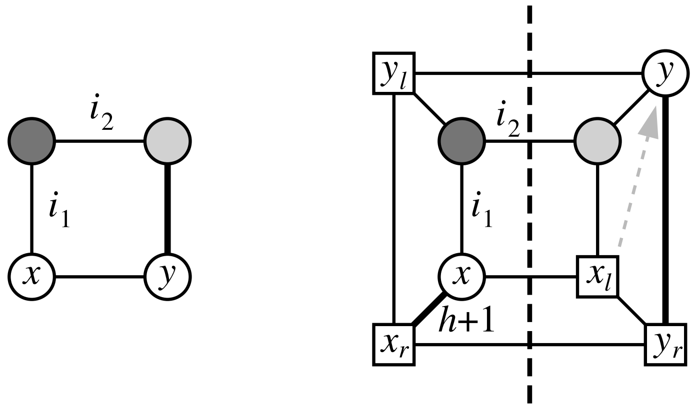

Figure 4: A DRCBT Spawning New Leaves in Its Optimal Hypercube

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 proof of this embedding. To proof the theorem, we will proof the following stronger claim on the embedding.

> *Given the embedding of the double-rooted complete binary tree of height h into its optimal hypercube. There exists a partition of the vertices of the double-rooted complete binary tree into 2^{h-2} groups of four vertices such that each group satisfy 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 two internal vertices is a parent of the right leaf,*
>
> *iii) the images of the leaf and its parent fulfilling condition ii) form a 1-dimensional subcube,*
>
> *iv) the images of the four vertices form a 2-dimensional subcube.*

Obviously the a double-rooted complete binary tree of height 2 can be embedded into the 2-dimensional hypercube satisfying the four conditions above. This can also be seen from the left part of Figure 4, where the white vertices are the leaves and the gray vertices are its parents. As induction hypothesis, the light gray vertex in Figure 4 is the parent of its leaf as required in condition *ii)*; the dark 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 locations 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 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 four conditions are satisfied by splitting the 3-dimensional subcube into a left and right 2-dimensional subcube as shown in Figure 4.

From the proof of the embedding, we will 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 dimension due to conditions *i) – iv)* and i_2 is the hypercube dimension connecting the two leaves. Note that the two leaves containing in the same subcube according to condition *iv)* have the same subcube information. In what follows, we denote by $\mathrm{loc}_h(v)$ the hypercube location of the vertex v in a h-dimensional subcube.

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

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

The hypercube locations of the double-rooted complete binary tree of height $h{+}1$ can be computed from the hypercube locations of the double-rooted complete binary tree of height h 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.

$$
\begin{array}{rclcrcl}
\mathrm{loc}_{h+1}(v) & = & \mathrm{loc}_h(v) \cdot 0 & \text{for all internal vertices } v & & & \\
\mathrm{loc}_{h+1}(x) & = & \mathrm{loc}_h(x) \cdot 0 & & & & \\
\mathrm{loc}_{h+1}(x_l) & = & (\mathrm{loc}_h(x))^{i_2} \cdot 0 & & \sigma(x_l) & = & (i_1, h+1) \\
\mathrm{loc}_{h+1}(x_r) & = & \mathrm{loc}_h(x) \cdot 1 & & \sigma(x_r) & = & (h+1, i_1) \\
\mathrm{loc}_{h+1}(y) & = & (\mathrm{loc}_h(y))^{i_1} \cdot 1 & & & & \\
\mathrm{loc}_{h+1}(y_l) & = & ((\mathrm{loc}_h(y))^{i_1})^{i_2} \cdot 1 & & \sigma(y_l) & = & (h+1, i_1) \\
\mathrm{loc}_{h+1}(y_r) & = & \mathrm{loc}_h(y) \cdot 1 & & \sigma(y_r) & = & (i_1, h+1)
\end{array}
$$

If the height of the double-rooted complete binary tree is known in advance or if we can determine when the last level is reached, we can clearly compute the final hypercube location of each newly spawned leaf. We will denote by h the maximal level, i.e., the height oft 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 as actually needed. This mapping is illustrated in Figure 5. From this illustration, we

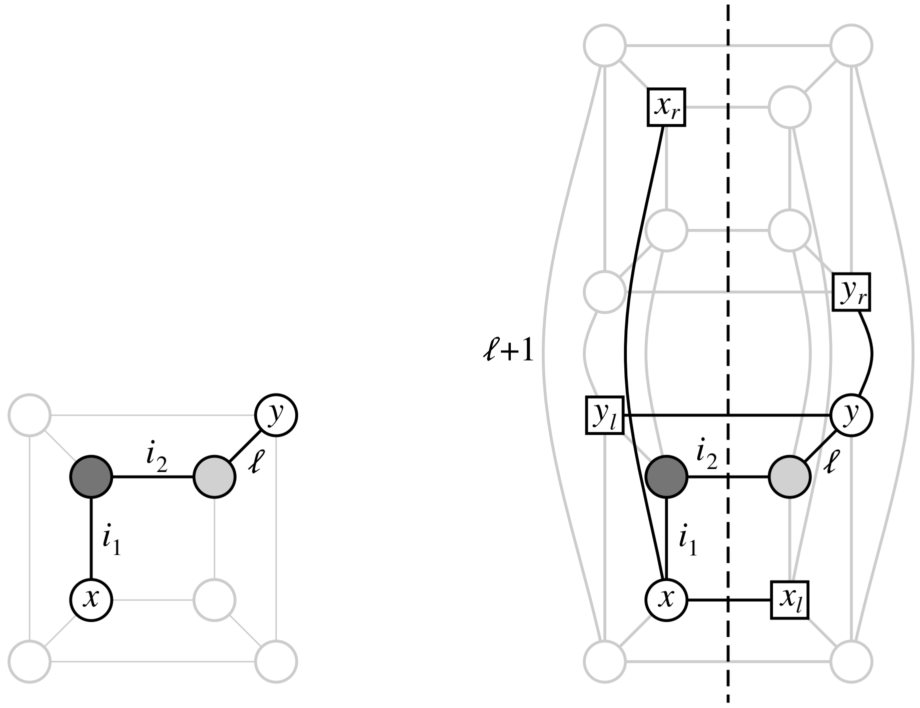

Figure 5: A DRCBT Spawning New Leaves at Level ℓ in a Larger Hypercube

obtain immediately the following construction.

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

Altogether, we have proved the following theorems.

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 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 only vertices

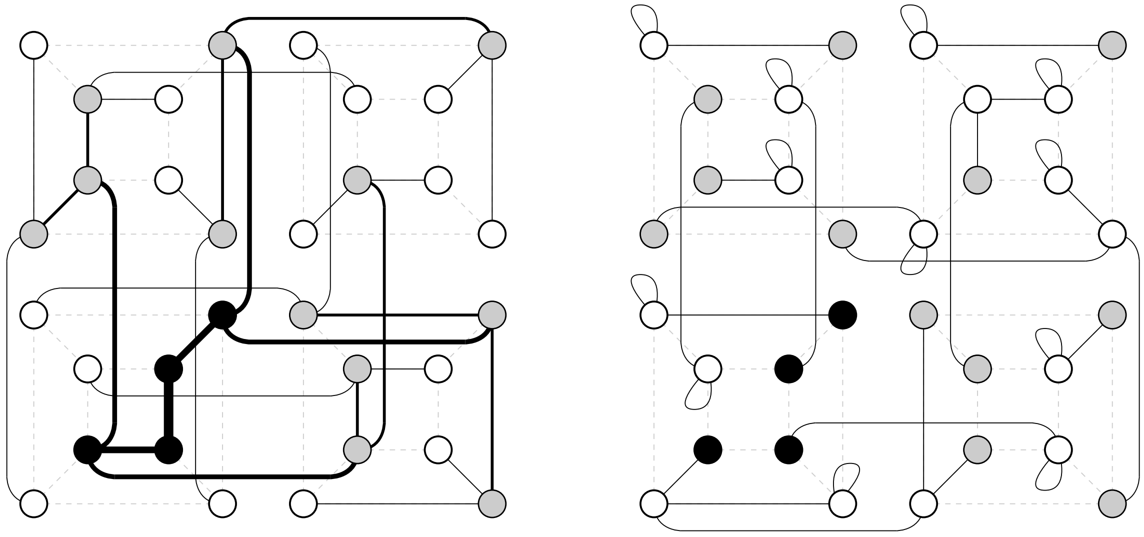

Figure 6: An Embedding of a DRCBT of Height 6 into a 5-dimensional Hypercube

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

A first simple approach to obtain a dynamic embedding of the double-rooted complete binary tree of height $h{+}1$ into a h-dimensional hypercube with unit dilation and load 2 is to embed the double-rooted complete binary tree of height h into a 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 location $\mathrm{loc}_h(v)$ and v_r to the hypercube location $(\mathrm{loc}_h(v))^{i_1}$, where $\sigma(v){=}(i_1, i_2)$. Thus, we get the following theorems.

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

Corollary 12 *A complete binary tree of height $h{+}1$ can be dynamically embedded into a h-dimensional hypercube with dilation 2, load 2 and congestion 2. Vertices of the same level are mapped to different hypercube locations. Moreover, all but one tree edges have unit dilation. The computation time for each new level is constant.*

With regard to the congestion, we can do even better. Using [RG93], we can find a dynamic embedding for double-rooted complete binary tree of height at least 6 into a hypercube with unit congestion. In [RG93], 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 which is illustrated in Figure 6. For convenience, the embedding of the first five levels was given in the left part of Figure 6 while the the embedding of the leaves are given in the right part of the Figure. The root path is indicated by black nodes while the parent of the leaves are drawn white. Note that the root path traverses three different dimensions as our four vertices in induction step of our embedding in the previous section. Thus, we obtain an embedding of a double-rooted complete binary tree into a hypercube with unit dilation, unit congestion and load 2, if we modify our embedding in the last four levels as given in Figure 6. Obviously, the embedding of these lower levels can be computed in constant time for each new level.

Theorem 13 *A double-rooted complete binary tree of height $h{+}1{\geq}6$ can be dynamically embedded into a h-dimensional hypercube with unit dilation, unit congestion, and load 2. Vertices of the same level are mapped to different hypercube locations. The computation time for each new level is constant.*

Clearly, this theorem is optimal. As mentioned in [RG93] it is not possible to improve the last theorem for trees of smaller height. 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 a h-dimensional hypercube has congestion 1 and every hypercube location 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 location as their parents.

Theorem 14 *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 $\ell{\in}[1{:}d{+}1]$ are mapped to different hypercube locations; vertices of a fixed level $\ell{\geq}d$ are distributed evenly among all hypercube locations. The computation time for each new level is constant.*

Corollary 15 *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 $\ell{\in}[1{:}d{+}1]$ are mapped to different hypercube locations; vertices of a fixed level $\ell{\geq}d$ are distributed evenly among all hypercube locations. Moreover, all but one tree edges have unit dilation. The computation time for each new level is constant.*

Sometimes, it might be useful if siblings are not mapped to the same hypercube locations. Again, the previous embedding can be extended such that the congestion is at most 2 as follows. As can be seen from Figure 7, we can embed 4 binary complete trees of height 2 into an 2-dimensional hypercube with load 2 and unit congestion. This can be used to extend our embedding as a new level of leaves grows such that each hypercube edge has congestion at most 2. It remains to compute the height of the double-rooted complete binary tree which can be embedded in this manor. The vertices of the double-rooted complete binary tree at level $h{+}1{+}\ell$ are distributed evenly among the hypercube locations, i.e., each hypercube locations is an image of exactly 2^{ℓ} tree vertices at level $h{+}1{+}\ell$. Since the required number of dimension is twice the load, we need $2{\cdot}2^{\ell}$ different

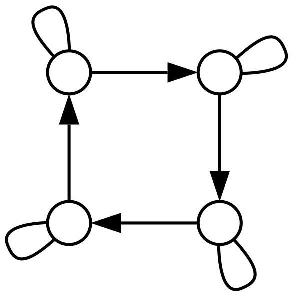

Figure 7: Embedding of 4 Trees of Height 2 with Congestion 1

dimensions for each new embedded level. Thus, the height of the tree which can be embedded in this manor is bounded by $h+2+k$, where k is the maximal value satisfying:

$$\sum_{\ell=0}^{k} 2^{\ell+1} \leq \frac{h}{2} 2^h.$$

This implies that $k=h-3+\lfloor \log(h) \rfloor$. Hence, we get the following theorems which are optimal.

Theorem 16 *A double-rooted complete binary tree of height $h+k$ can be dynamically embedded into a h-dimensional hypercube with unit dilation, congestion 2, and load 2^{k-1}, where $k \in [2{:}h-1+\lfloor \log(h) \rfloor]$. Vertices of a fixed level $\ell \in [1{:}h+1]$ are mapped to different hypercube locations; vertices of a fixed level $\ell \in [h+2{:}h+k]$ are distributed evenly among all hypercube locations. Also each pair of siblings is mapped to two different hypercube locations. The computation time for each new level is constant.*

Corollary 17 *A complete binary tree of height $h+k$ can be dynamically embedded into a h-dimensional hypercube with dilation 2, congestion 2, and load 2^{k-1}, where $k \in [2{:}h-1+\lfloor \log(h) \rfloor]$. Vertices of a fixed level $\ell \in [1{:}h+1]$ are mapped to different hypercube locations; vertices of a fixed level $\ell \in [h+2{:}h+k]$ are distributed evenly among all hypercube locations. Also each pair of siblings is mapped to two different hypercube locations. Moreover, all but one tree edges have unit dilation. The computation time for each new level is constant.*

6 Conclusion

We have presented some simple algorithms for dynamic embeddings of complete binary trees into hypercubes where the dilation, load, expansion and congestion are as small as possible. 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 [LNRS92]. Allowing randomization, a dynamic embedding of arbitrary binary trees into their optimal hypercubes with dilation 12, and with high probability constant load and constant congestion are constructed in [LNRS92]. 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.

References

[BI85] S. Bhatt, I. Ipsen: How to Embed Tress in Hypercubes, Technical Report DCS/RR-443, Department of Computer Science, Yale University, 1985.

[FM87] T. Feder, E. Mayr: An Efficient Algorithm for Embedding Complete Binary Trees in the Hypercube, Manuscript, Department of Computer Science, Stanford University, 1987.

[HL73] I. Havel, P. Liebl: Embedding the Polytomic Tree into the n-Cube, *Časopis. Pěst. Mat.*, Vol. 98 (1973), 307–314.

[LNRS92] T. Leighton, M. Newman, A. Ranade, W. Schwabe: Dynamic Tree Embeddings in Butterflies and Hypercubes, *SIAM Journal on Computing*, Vol. 21 (1992), No. 4, 639–654.

[LR91] E. Leiss, H. Reddy: Embedding Complete Binary Trees into Hypercubes, *Inf. Process. Lett.*, Vol. 38 (1991), 197–199.

[RG93] S. Ravindran, A. Gibbons: Dense Edge-Disjoint Embedding of Complete Binary Trees in the Hypercube, *Inf. Process. Lett.*, No. 6, Vol. 45 (1993), 321–325.

[SS85] Y. Saad, M. Schulz: Topological Properties of the Hypercube, Research Report RR-389, Department of Computer Science, Yale University, 1985.

[W94] A. Wagner: Embedding the Complete Tree in the Hypercube, *J. Parallel Distrib. Comput.*, No. 2, Vol. 20 (1994), 241–247.

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

342/1/90 A	Robert Gold, Walter Vogler: Quality Criteria for Partial Order Semantics of Place/Transition-Nets, Januar 1990
342/2/90 A	Reinhard Fößmeier: Die Rolle der Lastverteilung bei der numerischen Parallelprogrammierung, Februar 1990
342/3/90 A	Klaus-Jörn Lange, Peter Rossmanith: Two Results on Unambiguous Circuits, Februar 1990
342/4/90 A	Michael Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations-Mehrgitter-Methode
342/5/90 A	Reinhold Letz, Johann Schumann, Stephan Bayerl, Wolfgang Bibel: SETHEO: A High-Performance Theorem Prover
342/6/90 A	Johann Schumann, Reinhold Letz: PARTHEO: A High Performance Parallel Theorem Prover
342/7/90 A	Johann Schumann, Norbert Trapp, Martin van der Koelen: SETHEO/PARTHEO Users Manual
342/8/90 A	Christian Suttner, Wolfgang Ertel: Using Connectionist Networks for Guiding the Search of a Theorem Prover
342/9/90 A	Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Jaroslav Kremenek, Robert Lindhof, Thomas Ludwig, Peter Luksch, Thomas Treml: TOPSYS, Tools for Parallel Systems (Artikelsammlung)
342/10/90 A	Walter Vogler: Bisimulation and Action Refinement
342/11/90 A	Jörg Desel, Javier Esparza: Reachability in Reversible Free-Choice Systems
342/12/90 A	Rob van Glabbeek, Ursula Goltz: Equivalences and Refinement
342/13/90 A	Rob van Glabbeek: The Linear Time - Branching Time Spectrum
342/14/90 A	Johannes Bauer, Thomas Bemmerl, Thomas Treml: Leistungsanalyse von verteilten Beobachtungs- und Bewertungswerkzeugen
342/15/90 A	Peter Rossmanith: The Owner Concept for PRAMs
342/16/90 A	G. Böckle, S. Trosch: A Simulator for VLIW-Architectures
342/17/90 A	P. Slavkovsky, U. Rüde: Schnellere Berechnung klassischer Matrix-Multiplikationen
342/18/90 A	Christoph Zenger: SPARSE GRIDS

Reihe A

342/19/90 A	Michael Griebel, Michael Schneider, Christoph Zenger: A combination technique for the solution of sparse grid problems
342/20/90 A	Michael Griebel: A Parallelizable and Vectorizable Multi-Level-Algorithm on Sparse Grids
342/21/90 A	V. Diekert, E. Ochmanski, K. Reinhardt: On confluent semi-commutations-decidability and complexity results
342/22/90 A	Manfred Broy, Claus Dendorfer: Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions
342/23/90 A	Rob van Glabbeek, Ursula Goltz: A Deadlock-sensitive Congruence for Action Refinement
342/24/90 A	Manfred Broy: On the Design and Verification of a Simple Distributed Spanning Tree Algorithm
342/25/90 A	Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Peter Luksch, Roland Wismüller: TOPSYS - Tools for Parallel Systems (User's Overview and User's Manuals)
342/26/90 A	Thomas Bemmerl, Arndt Bode, Thomas Ludwig, Stefan Tritscher: MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual)
342/27/90 A	Wolfgang Ertel: Random Competition: A Simple, but Efficient Method for Parallelizing Inference Systems
342/28/90 A	Rob van Glabbeek, Frits Vaandrager: Modular Specification of Process Algebras
342/29/90 A	Rob van Glabbeek, Peter Weijland: Branching Time and Abstraction in Bisimulation Semantics
342/30/90 A	Michael Griebel: Parallel Multigrid Methods on Sparse Grids
342/31/90 A	Rolf Niedermeier, Peter Rossmanith: Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits
342/32/90 A	Inga Niepel, Peter Rossmanith: Uniform Circuits and Exclusive Read PRAMs
342/33/90 A	Dr. Hermann Hellwagner: A Survey of Virtually Shared Memory Schemes
342/1/91 A	Walter Vogler: Is Partial Order Semantics Necessary for Action Refinement?
342/2/91 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Rainer Weber: Characterizing the Behaviour of Reactive Systems by Trace Sets
342/3/91 A	Ulrich Furbach, Christian Suttner, Bertram Fronhöfer: Massively Parallel Inference Systems
342/4/91 A	Rudolf Bayer: Non-deterministic Computing, Transactions and Recursive Atomicity
342/5/91 A	Robert Gold: Dataflow semantics for Petri nets
342/6/91 A	A. Heise; C. Dimitrovici: Transformation und Komposition von P/T-Netzen unter Erhaltung wesentlicher Eigenschaften

Reihe A

342/7/91 A	Walter Vogler: Asynchronous Communication of Petri Nets and the Refinement of Transitions
342/8/91 A	Walter Vogler: Generalized OM-Bisimulation
342/9/91 A	Christoph Zenger, Klaus Hallatschek: Fouriertransformation auf dünnen Gittern mit hierarchischen Basen
342/10/91 A	Erwin Loibl, Hans Obermaier, Markus Pawlowski: Towards Parallelism in a Relational Database System
342/11/91 A	Michael Werner: Implementierung von Algorithmen zur Kompaktifizierung von Programmen für VLIW-Architekturen
342/12/91 A	Reiner Müller: Implementierung von Algorithmen zur Optimierung von Schleifen mit Hilfe von Software-Pipelining Techniken
342/13/91 A	Sally Baker, Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Udo Graf, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Rainer Knödlseder, Jaroslav Kremenek, Siegfried Langenbuch, Robert Lindhof, Thomas Ludwig, Peter Luksch, Roy Milner, Bernhard Ries, Thomas Treml: TOPSYS - Tools for Parallel Systems (Artikelsammlung); 2., erweiterte Auflage
342/14/91 A	Michael Griebel: The combination technique for the sparse grid solution of PDE's on multiprocessor machines
342/15/91 A	Thomas F. Gritzner, Manfred Broy: A Link Between Process Algebras and Abstract Relation Algebras?
342/16/91 A	Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Thomas Treml, Roland Wismüller: The Design and Implementation of TOPSYS
342/17/91 A	Ulrich Furbach: Answers for disjunctive logic programs
342/18/91 A	Ulrich Furbach: Splitting as a source of parallelism in disjunctive logic programs
342/19/91 A	Gerhard W. Zumbusch: Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme
342/20/91 A	M. Jobmann, J. Schumann: Modelling and Performance Analysis of a Parallel Theorem Prover
342/21/91 A	Hans-Joachim Bungartz: An Adaptive Poisson Solver Using Hierarchical Bases and Sparse Grids
342/22/91 A	Wolfgang Ertel, Theodor Gemenis, Johann M. Ph. Schumann, Christian B. Suttner, Rainer Weber, Zongyan Qiu: Formalisms and Languages for Specifying Parallel Inference Systems
342/23/91 A	Astrid Kiehn: Local and Global Causes
342/24/91 A	Johann M.Ph. Schumann: Parallelization of Inference Systems by using an Abstract Machine
342/25/91 A	Eike Jessen: Speedup Analysis by Hierarchical Load Decomposition
342/26/91 A	Thomas F. Gritzner: A Simple Toy Example of a Distributed System: On the Design of a Connecting Switch

Reihe A

342/27/91 A	Thomas Schnekenburger, Andreas Weininger, Michael Friedrich: Introduction to the Parallel and Distributed Programming Language ParMod-C
342/28/91 A	Claus Dendorfer: Funktionale Modellierung eines Postsystems
342/29/91 A	Michael Griebel: Multilevel algorithms considered as iterative methods on indefinite systems
342/30/91 A	W. Reisig: Parallel Composition of Liveness
342/31/91 A	Thomas Bemmerl, Christian Kasperbauer, Martin Mairandres, Bernhard Ries: Programming Tools for Distributed Multiprocessor Computing Environments
342/32/91 A	Frank Leßke: On constructive specifications of abstract data types using temporal logic
342/1/92 A	L. Kanal, C.B. Suttner (Editors): Informal Proceedings of the Workshop on Parallel Processing for AI
342/2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS
342/2-2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS - Revised Version (erschienen im Januar 1993)
342/3/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems
342/4/92 A	Claus Dendorfer, Rainer Weber: Development and Implementation of a Communication Protocol - An Exercise in FOCUS
342/5/92 A	Michael Friedrich: Sprachmittel und Werkzeuge zur Unterstützung paralleler und verteilter Programmierung
342/6/92 A	Thomas F. Gritzner: The Action Graph Model as a Link between Abstract Relation Algebras and Process-Algebraic Specifications
342/7/92 A	Sergei Gorlatch: Parallel Program Development for a Recursive Numerical Algorithm: a Case Study
342/8/92 A	Henning Spruth, Georg Sigl, Frank Johannes: Parallel Algorithms for Slicing Based Final Placement
342/9/92 A	Herbert Bauer, Christian Sporrer, Thomas Krodel: On Distributed Logic Simulation Using Time Warp
342/10/92 A	H. Bungartz, M. Griebel, U. Rüde: Extrapolation, Combination and Sparse Grid Techniques for Elliptic Boundary Value Problems
342/11/92 A	M. Griebel, W. Huber, U. Rüde, T. Störtkuhl: The Combination Technique for Parallel Sparse-Grid-Preconditioning and -Solution of PDEs on Multiprocessor Machines and Workstation Networks
342/12/92 A	Rolf Niedermeier, Peter Rossmanith: Optimal Parallel Algorithms for Computing Recursively Defined Functions

Reihe A

342/13/92 A	Rainer Weber: Eine Methodik für die formale Anforderungsspezifkation verteilter Systeme
342/14/92 A	Michael Griebel: Grid– and point–oriented multilevel algorithms
342/15/92 A	M. Griebel, C. Zenger, S. Zimmer: Improved multilevel algorithms for full and sparse grid problems
342/16/92 A	J. Desel, D. Gomm, E. Kindler, B. Paech, R. Walter: Bausteine eines kompositionalen Beweiskalküls für netzmodellierte Systeme
342/17/92 A	Frank Dederichs: Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen
342/18/92 A	Andreas Listl, Markus Pawlowski: Parallel Cache Management of a RDBMS
342/19/92 A	Erwin Loibl, Markus Pawlowski, Christian Roth: PART: A Parallel Relational Toolbox as Basis for the Optimization and Interpretation of Parallel Queries
342/20/92 A	Jörg Desel, Wolfgang Reisig: The Synthesis Problem of Petri Nets
342/21/92 A	Robert Balder, Christoph Zenger: The d-dimensional Helmholtz equation on sparse Grids
342/22/92 A	Ilko Michler: Neuronale Netzwerk-Paradigmen zum Erlernen von Heuristiken
342/23/92 A	Wolfgang Reisig: Elements of a Temporal Logic. Coping with Concurrency
342/24/92 A	T. Störtkuhl, Chr. Zenger, S. Zimmer: An asymptotic solution for the singularity at the angular point of the lid driven cavity
342/25/92 A	Ekkart Kindler: Invariants, Compositionality and Substitution
342/26/92 A	Thomas Bonk, Ulrich Rüde: Performance Analysis and Optimization of Numerically Intensive Programs
342/1/93 A	M. Griebel, V. Thurner: The Efficient Solution of Fluid Dynamics Problems by the Combination Technique
342/2/93 A	Ketil Stølen, Frank Dederichs, Rainer Weber: Assumption / Commitment Rules for Networks of Asynchronously Communicating Agents
342/3/93 A	Thomas Schnekenburger: A Definition of Efficiency of Parallel Programs in Multi-Tasking Environments
342/4/93 A	Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: A Proof of Convergence for the Combination Technique for the Laplace Equation Using Tools of Symbolic Computation
342/5/93 A	Manfred Kunde, Rolf Niedermeier, Peter Rossmanith: Faster Sorting and Routing on Grids with Diagonals
342/6/93 A	Michael Griebel, Peter Oswald: Remarks on the Abstract Theory of Additive and Multiplicative Schwarz Algorithms
342/7/93 A	Christian Sporrer, Herbert Bauer: Corolla Partitioning for Distributed Logic Simulation of VLSI Circuits

Reihe A

342/8/93 A	Herbert Bauer, Christian Sporrer: Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving
342/9/93 A	Peter Slavkovsky: The Visibility Problem for Single-Valued Surface (z=$f(x, y)$): The Analysis and the Parallelization of Algorithms
342/10/93 A	Ulrich Rüde: Multilevel, Extrapolation, and Sparse Grid Methods
342/11/93 A	Hans Regler, Ulrich Rüde: Layout Optimization with Algebraic Multigrid Methods
342/12/93 A	Dieter Barnard, Angelika Mader: Model Checking for the Modal Mu-Calculus using Gauß Elimination
342/13/93 A	Christoph Pflaum, Ulrich Rüde: Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids
342/14/93 A	Christoph Pflaum: Convergence of the Combination Technique for the Finite Element Solution of Poisson's Equation
342/15/93 A	Michael Luby, Wolfgang Ertel: Optimal Parallelization of Las Vegas Algorithms
342/16/93 A	Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: Pointwise Convergence of the Combination Technique for Laplace's Equation
342/17/93 A	Georg Stellner, Matthias Schumann, Stefan Lamberts, Thomas Ludwig, Arndt Bode, Martin Kiehl und Rainer Mehlhorn: Developing Multicomputer Applications on Networks of Workstations Using NXLib
342/18/93 A	Max Fuchs, Ketil Stølen: Development of a Distributed Min/Max Component
342/19/93 A	Johann K. Obermaier: Recovery and Transaction Management in Write-optimized Database Systems
342/20/93 A	Sergej Gorlatch: Deriving Efficient Parallel Programs by Systemating Coarsing Specification Parallelism
342/01/94 A	Reiner Hüttl, Michael Schneider: Parallel Adaptive Numerical Simulation
342/02/94 A	Henning Spruth, Frank Johannes: Parallel Routing of VLSI Circuits Based on Net Independency
342/03/94 A	Henning Spruth, Frank Johannes, Kurt Antreich: PHIroute: A Parallel Hierarchical Sea-of-Gates Router
342/04/94 A	Martin Kiehl, Rainer Mehlhorn, Matthias Schumann: Parallel Multiple Shooting for Optimal Control Problems Under NX/2
342/05/94 A	Christian Suttner, Christoph Goller, Peter Krauss, Klaus-Jörn Lange, Ludwig Thomas, Thomas Schnekenburger: Heuristic Optimization of Parallel Computations
342/06/94 A	Andreas Listl: Using Subpages for Cache Coherency Control in Parallel Database Systems

Reihe A

342/07/94 A	Manfred Broy, Ketil Stølen: Specification and Refinement of Finite Dataflow Networks - a Relational Approach
342/08/94 A	Katharina Spies: Funktionale Spezifikation eines Kommunikationsprotokolls
342/09/94 A	Peter A. Krauss: Applying a New Search Space Partitioning Method to Parallel Test Generation for Sequential Circuits
342/10/94 A	Manfred Broy: A Functional Rephrasing of the Assumption/Commitment Specification Style
342/11/94 A	Eckhardt Holz, Ketil Stølen: An Attempt to Embed a Restricted Version of SDL as a Target Language in Focus
342/12/94 A	Christoph Pflaum: A Multi-Level-Algorithm for the Finite-Element-Solution of General Second Order Elliptic Differential Equations on Adaptive Sparse Grids
342/13/94 A	Manfred Broy, Max Fuchs, Thomas F. Gritzner, Bernhard Schätz, Katharina Spies, Ketil Stølen: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems
342/14/94 A	Maximilian Fuchs: Technologieabhängigkeit von Spezifikationen digitaler Hardware
342/15/94 A	M. Griebel, P. Oswald: Tensor Product Type Subspace Splittings And Multilevel Iterative Methods For Anisotropic Problems
342/16/94 A	Gheorghe Ştefănescu: Algebra of Flownomials
342/17/94 A	Ketil Stølen: A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers
342/18/94 A	Michael Griebel, Tilman Neuhoeffer: A Domain-Oriented Multilevel Algorithm-Implementation and Parallelization
342/19/94 A	Michael Griebel, Walter Huber: Turbulence Simulation on Sparse Grids Using the Combination Method
342/20/94 A	Johann Schumann: Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study -
342/01/95 A	Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids
342/02/95 A	Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law
342/03/95 A	Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space
342/04/95 A	Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes
342/05/95 A	Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems
342/06/95 A	Maximilian Fuchs: Formal Design of a Model-N Counter

Reihe A

342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology

342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies

342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness

342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design

342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System

342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm

342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming

342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks

342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik

342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations

342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications

342/19/95 A Thomas Schnekenburger: Integration of Load Distribution into ParMod-C

342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

342/21/95 A Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control

342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes

342/23/95 A Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation

342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems

342/01/96 A Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries

342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs

Reihe A

342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B	Wolfgang Reisig: Petri Nets and Algebraic Specifications
342/2/90 B	Jörg Desel: On Abstraction of Nets
342/3/90 B	Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems
342/4/90 B	Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme
342/1/91 B	Barbara Paech1: Concurrency as a Modality
342/2/91 B	Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier-Toolbox-Anwenderbeschreibung
342/3/91 B	Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen
342/4/91 B	Werner Pohlmann: A Limitation of Distributed Simulation Methods
342/5/91 B	Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis
342/6/91 B	Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme
342/7/91 B	W. Reisig: Concurrent Temporal Logic
342/1/92 B	Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support
	Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support
342/2/92 B	Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen
342/1/93 B	Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung
342/2/93 B	Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick
342/1/94 B	Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS