

ON EMBEDDING BINARY TREES INTO HYPERCUBES *

Woei-Kae Chen

Department of Electronic Engineering
National Taipei Institute of Technology
1. Section 3, Chung-Hsiao East Road
Taipei, Taiwan
Republic of China

Matthias F.M. Stallmann

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206

Written: September, 1990.

Revised: July 23, 1997

To appear in *Journal on Parallel and Distributed Computing*.

Please direct all correspondence to Prof. Stallmann at the above address.

* This research was partially supported by the Office of Naval Research under contract N00014-88-K-0555, which is gratefully acknowledged.

Proofs should be sent to:

Matthias Stallmann

Department of Computer Science

North Carolina State University

Raleigh, NC 27695-8206

phone: (919) 515-7978

email: `Matt_Stallmann@ncsu.edu`

Abstract. Hypercubes are known to be able to simulate other structures such as grids and binary trees. It has been shown that an arbitrary binary tree can be embedded into a hypercube with constant expansion and constant dilation. This paper presents a simple linear-time heuristic which embeds an arbitrary binary tree into a hypercube with expansion 1 and average dilation no more than 2. We also give some results extending good embeddings for parity-balanced binary trees to arbitrary binary trees. In particular, we show that a conjecture of Havel [8] implies embeddings of binary trees into hypercubes with expansion 1 and either dilation 2 or average dilation approaching 1, and embeddings with expansion 2 and dilation 1.

Key words. Hypercubes, hypercube embedding, mapping problem, binary trees.

List of symbols:

O oh

0 zero

x ex

k kay

1 one

l ell

v vee

n_e en sub ee

n_o en sub (little) oh

\hat{T} tee hat

T' tee prime

T'' tee double prime

1. Introduction. The hypercube embedding problem, a restricted version of the general mapping problem, is the problem of mapping a communication graph into a hypercube multiprocessor. A k -cube is an undirected graph $H = (V_H, E_H)$ consisting of $n = 2^k$ vertices labeled from 0 to $n - 1$, such that there is an edge between any two vertices if and only if the binary representation of their labels differs by exactly one bit. An *embedding* f of a graph $G = (V, E)$ into a k -cube $H = (V_H, E_H)$ is a one-to-one function $f : V \mapsto V_H$. For an embedding f , the dilation cost of an edge $\{v, w\} \in E$ is $d(f(v), f(w))$, i.e., the Hamming distance between $f(v)$ and $f(w)$. For synchronous algorithms, the quality of an embedding f is usually evaluated by its *maximum dilation* $\max_{\{v, w\} \in E} d(f(v), f(w))$ (also called simply the *dilation* of f). For asynchronous algorithms, the *average dilation* $(1/|E|) \cdot \sum_{\{v, w\} \in E} d(f(v), f(w))$ is usually used (see [7] for comments on using average dilation as a cost measure). In addition to the dilation cost, the *expansion* cost $\lceil |V_H| / |V| \rceil$ is used to evaluate the processor utilization (since a k -cube has exactly 2^k vertices, the expansion cost of embedding an n -vertex communication graph into a $\lceil \log n \rceil$ -cube is 1).

Binary trees are an important class of communication graphs since they are a common data structure and are natural to divide-and-conquer algorithms. A binary tree is a tree in which the degree of every vertex is less than or equal to 3. The embedding of complete binary trees has been studied in [11, 13]. A complete binary tree with $2^k - 1$ vertices is not a subgraph of a k -cube [13]. However, Nebeský [11] showed that a complete binary tree can be embedded into a hypercube with either expansion 2 and dilation 1 or with expansion 1 and dilation 2 (in fact, only one edge in the complete binary tree is embedded in dilation 2).

Afrati et al. [1] describe a divide-and-conquer algorithm that gives dilation-1 embeddings in a hypercube of dimension at most dk for trees having at most 2^k nodes and a maximum degree of d . In the case of binary trees, this algorithm embeds an n -node binary tree in a hypercube of at most $O(n^{1.71})$ nodes (based on the fact that removal of one edge suffices to separate a binary tree into two subtrees, each having no more than $2/3$ of the nodes). Wagner [12] improved this result by showing that any binary tree can be embedded into an $O(n \log n)$ -node hypercube in dilation 1.

Bhatt et al. [4] first showed that any arbitrary binary tree can be embedded in a hypercube with $O(1)$ dilation and $O(1)$ expansion (the embedding also has $O(1)$ congestion, a measure not discussed here). The constant factor of this embedding is too large to make it of practical interest. Monien and Sudborough [10] improved the result by giving an embedding of dilation 3 and expansion $O(1)$ and an embedding of dilation 5 and expansion 1.

Obviously, an embedding heuristic that guarantees a constant dilation also guarantees a constant average dilation. However, heuristics such as [2], [4], and [10] were not designed with the average dilation metric in mind, and the average dilations obtained by them are not likely to be competitive. Also, it is desirable to have a simpler heuristic so that implementation is easier. This paper presents a simple linear-time embedding strategy called the preorder-Gray-code embedding. We will show that this strategy embeds arbitrary binary trees into hypercubes with expansion 1 and average dilation no more than 2.

Havel [8] conjectured that all *parity-balanced binary trees* with 2^k vertices can be embedded perfectly (i.e., with dilation 1 and expansion 1) in a k -cube (the conjecture

has been proven for the special case of caterpillars [9]). A parity-balanced tree is a tree in which the two bipartitions of the tree have the same number of vertices (recall that trees are bipartite; parity-balanced trees are also called *color-balanced trees*). For brevity's sake, we simply use “balanced” instead of “parity-balanced” in the remainder of the text. Readers should note that the “balanced trees” used here are not the “height-balanced trees” commonly used as efficient data structures for search and update problems.

A binary tree with 2^k vertices is perfectly embeddable in a k -cube only if it is balanced, since a hypercube has two balanced bipartitions. Thus Havel's conjecture states that for a 2^k -vertex binary tree to be perfectly embeddable in a k -cube, balance is not only a necessary but also a sufficient condition. In this paper, we show that good embeddings of balanced binary trees can be extended to good embeddings of arbitrary binary trees. In particular, we show that Havel's conjecture implies embeddings of binary trees into hypercubes with expansion 1 and either dilation 2 or average dilation approaching 1. Havel's conjecture also yields embeddings of binary trees with expansion 2 and dilation 1.

For simplicity we will use n as the number of vertices of the binary tree to be embedded. We assume, without loss of generality, that binary trees are rooted — any vertex with degree less than 3 can be the root. For an interior vertex v of a binary tree, the two children of v are called the left child, $L(v)$, and the right child, $R(v)$.

Section 2 describes the preorder-Gray-code embedding and proves an upper bound on its average dilation. Section 3 gives a lower bound on the average dilation of the preorder-Gray-code embedding. Section 4 shows that good embeddings (with respect to average dilation) for balanced binary trees can be extended to arbitrary binary trees.

Section 5 shows that embeddings for balanced binary trees can be extended to arbitrary binary trees with only a slight penalty in expansion or dilation. Conclusions are given in Section 6.

2. Preorder-Gray-Code Embedding. The binary reflected Gray code is well known and is widely used in digital systems because it is cyclic and the binary number system can easily be translated to the Gray code. For the hypercube embedding problem, the Gray code has a few useful properties that can be used to provide upper and lower bounds on the embedding distance. The k -bit Gray code sequence G_k is defined recursively as:

$$\begin{cases} G_1 &= (0, 1) \\ G_k &= (0G_{k-1}, 1G_{k-1}^R) \end{cases}$$

where G_{k-1}^R is the reverse of G_{k-1} .

For example, $G_2 = (00_2, 01_2, 11_2, 10_2) = (0, 1, 3, 2)$. Since G_k is constructed by reflecting two G_{k-1} 's, the Gray code is called a reflected code. It is useful to identify the mapping between the ordinary number system and the Gray code sequence. By definition, the number 0 corresponds to the first element of a G_k and the number i corresponds to the $(i + 1)$ th element of a G_k . We will use $g(x)$ to denote the Gray code corresponding to the number x . For example in G_2 , $g(0) = 0$, $g(1) = 1$, $g(2) = 3$, and $g(3) = 2$. It is well known that $g(x) = 2x \oplus x$, where \oplus is the bitwise exclusive-or operation. Thus $g(x)$ can be computed efficiently.

What follows is a list of important properties of the Gray code, used in this paper. They can all be proved directly from the definition or by straightforward induction.

Readers who are interested in the proofs may refer to [5] for details.

FACT 1. (*cyclic*) *The Gray code sequence is cyclic, i.e., for any G_k , $d(g(x), g(y)) = 1$, where $0 \leq x \leq 2^k - 1$ and $y = (x + 1) \bmod 2^k$.*

By definition, given a G_k and an $m \leq k$, G_k is constructed from alternating disjoint G_m 's and G_m^R 's, called *m-blocks*, with 2^{k-m-1} disjoint G_m 's and 2^{k-m-1} disjoint G_m^R 's. Note that within each *m-block*, the most significant $k - m$ bits of the code stay fixed while the remaining m bits cycle through all possibilities.

FACT 2. *In G_k the most significant $k - m$ bits of any two consecutive m-blocks differ by at most one bit.*

FACT 3. *Given G_k , $d(g(x), g(y)) = 2$, where $0 \leq x \leq 2^k - 1$ and $y = (x + 2^m) \bmod 2^k$ for some m with $1 \leq m \leq k - 1$.*

FACT 4. *Given G_k , $d(g(x), g(y)) \leq \lceil \log(|x - y|) \rceil + 1$ where $0 \leq x \leq 2^k - 1$ and $0 \leq y \leq 2^k - 1$.*

FACT 5. *Given G_k with $k \geq 3$, there exists, for any x and y , a z with $y \leq z \leq y + 5$ such that $d(g(x), g(z)) \geq 3$, where $0 \leq x \leq 2^k - 1$, $0 \leq y \leq 2^k - 1$, and $x < y$.*

The recursive preorder-Gray-code embedding algorithm is given in Figure 1. In the algorithm, $s(v)$ is the size of the subtree rooted at v , that is $s(v)$ equals the total number of descendants of v (including v). To execute the algorithm, the global variable i should be initialized to 0 and the root of the target binary tree should be passed as the parameter (the root can be arbitrarily chosen as any vertex with degree less than 3). What the algorithm does is label the vertices of the binary tree (i.e., assign $f(v)$ for each v) by the Gray codes of the preorder numbering. The preorder numbering used in this algorithm has the feature that smaller subtrees are always numbered first. When

the labeling is done, each vertex of the binary tree is assigned to the hypercube vertex corresponding to its label. The complexity of the algorithm is obviously linear.

A preorder-Gray-code embedding of an 8-vertex binary tree is shown in Figure 2. By Fact 1, the edges $\{a, b\}$, $\{b, c\}$, $\{c, e\}$, $\{d, f\}$, and $\{g, h\}$ are known immediately to be embedded in dilation 1. In fact, because of preorder numbering, v and $L(v)$ are numbered by consecutive numbers; thus at least a half of the edges of the binary tree are embedded in dilation 1. The dilations of the $\{v, R(v)\}$ edges are not known exactly in general. However, Fact 4 provides an upper bound for them. For example, the edge $\{b, d\}$ is embedded in dilation $d(g(1), g(4)) \leq \lceil \log(3) \rceil + 1 = 3$. Note that in this example, $\{b, d\}$ is actually embedded in dilation 3, the upper bound. Observe that the vertex d is numbered after vertices c and e (the entire left subtree of the vertex b), thus it is advantageous for the left subtree to be small. In the algorithm, the preorder numbering always numbers the smaller subtree first so that the dilations of the edges $\{v, R(v)\}$ are bounded. This observation is essential to the proof of the following theorem.

THEOREM 1. *The preorder-Gray-code embedding strategy embeds arbitrary binary trees into hypercubes with expansion 1 and average dilation no more than 2.*

Proof. Let T ($|T| = n$) be the binary tree to be embedded with root v . Let $l = s(L(v))$ and $r = s(R(v))$, so $n = l + r + 1$. The preorder numbers of v and $L(v)$ differ by 1 and the preorder numbers of v and $R(v)$ differ by $l + 1$. Thus, by Facts 1 and 4, the dilation of the edge $\{v, L(v)\}$ is always 1 and the dilation of the edge $\{v, R(v)\}$ is less than or equal to $\lceil \log(l + 1) \rceil + 1$. Since the smaller subtree is always numbered first in the preorder numbering, we are assured that $l \leq (n - 1)/2$ and $r \geq (n - 1)/2$.

The subtrees rooted at $L(v)$ and $R(v)$ are embedded using the same strategy, thus the total dilation $D(n)$ of the preorder-Gray-code embedding can be expressed by the recurrence:

$$\begin{cases} D(1) = 0 \\ D(n) \leq D(r) + 1 & \text{if } l = 0 \\ D(n) \leq D(l) + D(r) + 1 + (\lceil \log(l+1) \rceil + 1) & \text{if } l \neq 0 \end{cases}$$

We now show by induction that $D(n) \leq 2n - \lceil \log(n+1) \rceil - 1$. Thus the average dilation ≤ 2 . The hypothesis obviously holds for $n = 1$. Assuming $D(1), \dots, D(n-1)$ satisfy the hypothesis, we show that $D(n)$ also satisfies the hypothesis.

$$\begin{aligned} D(n) &\leq D(l) + D(r) + 1 + (\lceil \log(l+1) \rceil + 1) \\ &\leq 2l - \lceil \log(l+1) \rceil - 1 + 2r - \lceil \log(r+1) \rceil - 1 + \lceil \log(l+1) \rceil + 2 \\ &= 2(l+r) - \lceil \log(r+1) \rceil \\ &\leq 2n - \lceil \log(\frac{n-1}{2} + 1) \rceil - 2 \\ &= 2n - \lceil \log(\frac{n+1}{2}) \rceil - 2 \\ &= 2n - \lceil \log(n+1) \rceil - 1 \end{aligned}$$

This concludes the induction argument. \square

3. Lower bound. In the previous section, we have shown that the preorder-Gray-code embedding strategy embeds arbitrary binary trees into hypercubes with expansion 1 and average dilation no more than 2. This upper bound is an overestimate since it is based on Fact 4, a pessimistic upper bound on the Hamming distance of two gray

codes. In this section, we show that the lower bound on the average dilation of the preorder-Gray-code embedding is at least 1.78. More precisely, there exists an n -vertex binary tree such that the preorder-gray-code embedding embeds the tree with average dilation at least 1.78 for all sufficiently large n .

Based on the proof of the upper bound on average dilation, we know that the edges $\{v, R(v)\}$ will be embedded in a larger dilation if the size of the left subtree is larger. Therefore the preorder-Gray-code embedding does a poor job of embedding binary trees in which the sizes of the two subtrees are about the same. A complete binary tree constitutes the extreme case of this. However, it is not a worst-case binary tree for the preorder-Gray-code embedding. Observe that the preorder numbers of all v and $R(v)$ pairs in a complete binary tree differ by exact powers of 2, and by Fact 3, all v and $R(v)$ pairs are embedded in dilation 2. Thus the overall average dilation is 1.5 for complete binary trees.

To prove the lower bound, our strategy is to show that we can always construct an n -vertex binary tree ($n > 2^k$) from a set of smaller subtrees (the basis) whose size ranges from $2^{k-1} - 5$ to 2^k vertices, such that the constructed binary tree has an average dilation worse than the minimum average dilation of the basis. The basis is then verified empirically, using a greedy strategy for constructing bad examples, to show that the claimed lower bound is true.

Given interior vertex v it is desirable to have the dilation of the edge $\{v, R(v)\}$ as large as possible. However, since the upper bound on average dilation is 2 and the dilation of the edge $\{v, L(v)\}$ is 1, it suffices to make the edge $\{v, R(v)\}$ dilation 3. From Fact 5 it follows that we can always choose the size of the left subtree l such that

$l \geq \lceil (n-1)/2 \rceil - 5$, and the dilation of the edge $\{v, R(v)\}$ is at least 3. What this means is that we can always construct a large binary tree T with root v by combining two subtrees rooted at $L(v)$ and $R(v)$ each with roughly a half of the vertices of T . The overall average dilation of T must be greater than the minimum average dilation of the subtrees, since the two top-level edges $\{v, L(v)\}$ and $\{v, R(v)\}$ have an average dilation 2, the upper bound. The subtrees can be constructed using the same strategy recursively until their sizes fall within the range of the basis case.

We chose $k = 10$ for our basis, that is, we verified the lower bound for trees with 507 to 1024 vertices occurring in every possible Gray-code position. A larger basis (with a larger number of vertices) can be used to obtain a slightly better lower bound; however, it is much more costly to verify.

It is not too hard to verify the basis when $k = 10$; what needs to be shown is that for every size (507 to 1024) and every possible beginning position $\{g(x) \mid x \leq n\}$, there exists a binary tree such that the preorder-Gray-code embedding strategy embeds this binary tree with average dilation at least C (in this case $C = 1.78$). It may initially seem impossible to verify all possible beginning positions, since n can be arbitrarily large. However, since the Gray code is symmetric, only a subset of possible positions needs to be verified. If we let $m = 10$ in Fact 2, the Gray code sequence repeats every $2^m = 1024$ codes, thus for binary trees with ≤ 1024 vertices, we only have to verify the first 1024 possible positions. The strategy that we used to construct binary trees for the basis is a greedy search for the position that gives the worst dilation for the edge $\{v, R(v)\}$. This strategy results in a minimum average dilation for the basis of about 1.78. Thus the following theorem is established.

THEOREM 2. *For all $n \geq n_0$, there exists an n -vertex binary tree T such that the average dilation of the preorder-Gray-code embedding of T into a $\lceil \log n \rceil$ -cube is at least C (with $n_0 = 507$, we have $C = 1.78$).*

As was pointed out earlier, minor improvements to this bound are possible if a larger basis is chosen. More significant improvements would require a better strategy for building bad examples in the basis cases. The upper bound proved in Theorem 1 is probably a slight overestimate; we believe that the true upper bound for preorder-Gray-code embeddings is close to 1.9.

4. Balanced binary trees and average dilation. Recall that a balanced binary tree is a binary tree in which the two bipartitions of the tree have the same number of vertices. The two bipartitions of a tree are referred to as even and odd. A vertex in the even (odd) bipartition is called an even (odd) vertex. In general we will assume that there are more even vertices than odd vertices in an unbalanced binary tree. The following lemma is an observation on the leaves of an unbalanced tree (not necessarily binary) which leads to several interesting results related to Havel's conjecture.

LEMMA 3. *A tree with n_e even vertices and n_o odd vertices has at least $n_e - n_o + 1$ even leaves.*

Proof. Let x be the number of even leaves. Since the degree of a leaf is 1 and the degree of an interior vertex is at least 2, the sum of the degrees of all even vertices is $\text{degree}(\text{even}) \geq 2(n_e - x) + x = 2n_e - x$. But $\text{degree}(\text{even}) = n_e + n_o - 1$, the total number of edges in the tree. So $n_e + n_o - 1 \geq 2n_e - x$ and $x \geq n_e - n_o + 1$. \square

Two immediate consequences of Lemma 3 are (a) a balanced tree has at least one even leaf and at least one odd leaf, and (b) an unbalanced tree has at least one leaf

that belongs to the larger bipartition. These two results establish the equivalence of the embedding of balanced binary forests and balanced binary trees stated in the next lemma. Note that it is necessary to distinguish trees from forests for the hypercube embedding problem because embedding individual trees of forests one by one does not necessarily lead to an embedding as good as what could be obtained for the forest. This is different from most graph problems (e.g., coloring) where the solution for a forest can be derived directly from that on the trees.

LEMMA 4. *For every n , all n -vertex balanced binary forests can be embedded into hypercubes (with a given dilation and expansion) if and only if all n -vertex balanced binary trees can be embedded into hypercubes (with the same dilation and expansion).*

Proof. It is obvious that if all balanced binary forests are embeddable, all balanced binary trees are embeddable. We will show that one can always connect the components of a balanced binary forest so that the forest becomes a balanced binary tree. Thus if all balanced binary trees are embeddable, all balanced binary forest are also embeddable. To connect a balanced binary forest, we can iteratively join two components, one with an even leaf, another with an odd leaf, until the forest becomes a single connected component, a tree. Such components must exist because of (a) and (b) above. To join the two components, simply add an extra edge between the even leaf and the odd leaf. This process results in a balanced binary tree since each new edge connects an even leaf to an odd leaf which preserves the bipartition and the structure (acyclic, connected, and degree ≤ 3) of a balanced binary tree. \square

The next lemma is also known as the two color bisector theorem, and is used in VLSI layout applications. We use it to help convert arbitrary binary trees to balanced

binary forests.

LEMMA 5. (*Two-color bisector theorem*) *An n -vertex binary tree with b black vertices and w white vertices ($b + w = n$) can be bisected, with removal of no more than $2 \log n$ edges, into two subtrees each of size at least $\lfloor n/2 \rfloor$, and such that each contains at least $\lfloor b/2 \rfloor$ black and $\lfloor w/2 \rfloor$ white vertices.*

Proof. See [3]. \square

Given an n -vertex binary tree with n_e even vertices and n_o odd vertices, we can convert this binary tree to a balanced binary forest with the removal of no more than $O(\log n)$ edges as follows. First, color all even vertices black and all odd vertices white. By Lemma 5, the binary tree can be bisected into two partitions A and B each with at least $\lfloor n_e/2 \rfloor$ even vertices and $\lfloor n_o/2 \rfloor$ odd vertices. Now we can reverse the bipartition of all vertices of the partition A so that the partition A has $\lfloor n_e/2 \rfloor$ odd vertices and $\lfloor n_o/2 \rfloor$ even vertices. That is, the binary forest is balanced.

LEMMA 6. *If all balanced binary trees can be embedded into hypercubes with expansion e and average dilation C , then all binary trees can be embedded into hypercubes with expansion e and average dilation $C + O((\log n)^2/n)$.*

Proof. Given a binary tree T with n vertices, simply convert T into a balanced binary forest T' with the removal of $O(\log n)$ edges. By Lemma 4, T' can be embedded with expansion e and average dilation no more than C . Apply the embedding of T' to T . There are at most $O(\log n)$ edges (the removed edges) embedded in unknown dilation. Since the diameter of the target hypercube for the embedding of T' is at most $\lceil \log(en) \rceil$ and e is at most $O(\log n)$ [12], T can be embedded with average dilation $C + O(\log(en) \log n/n) = C + O((\log n)^2/n)$. \square

If Havel's conjecture is true, Lemma 6 implies that all arbitrary binary trees can be embedded into hypercubes with expansion 1 and average dilation $1 + O((\log n)^2/n)$. Thus we propose the following conjecture.

CONJECTURE. *An arbitrary n -vertex binary tree can be embedded into a hypercube with expansion 1 and average dilation approaching 1 as $n \rightarrow \infty$.*

5. Balanced binary trees and dilation. We show in this section how arbitrary binary trees can be transformed into balanced binary trees with minimal expansion or with minimal dilation. Thus good embeddings for balanced binary trees can be extended to obtain good embeddings for arbitrary binary trees. The two main results are Lemma 7 and Lemma 10, which show that good embeddings for balanced binary trees can be extended to all binary trees with only a slight penalty in expansion or dilation, respectively.

An immediate consequence of Lemma 3 is that we can always balance a binary tree by adding extra leaves to the smaller bipartition and the number of leaves needed is exactly the difference between the two bipartitions. Thus a binary tree T with n vertices can be balanced by adding no more than n vertices. The following lemma states that with a slight penalty on expansion, good embeddings of balanced binary trees can be extended to arbitrary binary trees. Note that a $(k+1)$ -cube has twice as many vertices as a k -cube, so the result in the lemma is the best possible.

LEMMA 7. *If, for all k , any 2^k -vertex balanced binary tree can be embedded into hypercubes with expansion e and dilation d , then, for all k , any 2^k -vertex binary tree can be embedded into hypercubes with expansion $2e$ and dilation d .*

Proof. Given a 2^k -vertex arbitrary binary tree T , simply transform T into a bal-

anced binary tree T' with at most 2^{k+1} vertices. Since T' is balanced, T' can be embedded into hypercubes with expansion e and dilation d . That is T' and thus T (a subtree) can be embedded into a $(2e \cdot 2^k)$ -vertex hypercube with dilation d . \square

For binary trees having an arbitrary number of vertices, the resulting expansion is $2e + 1$ instead of $2e$. For example, a 5-vertex T yields a 10-vertex T' , which requires a 16-vertex cube with expansion 1, so the expansion for T is 3. However, the expansion can be improved to $\lfloor 3e/2 \rfloor + 1$ with the observation that $n/2 - 1$ vertices suffice to balance an unbalanced tree (use Lemma 8, below, and the fact that at most $n/2 - 1$ vertices of an n -vertex binary tree can have degree 3).

We now show how an arbitrary binary tree T can be transformed into a balanced binary tree \hat{T} with the same number of vertices so that adjacent vertices in T are at most distance 2 away in \hat{T} . Given a binary tree T with n_e even vertices and n_o odd vertices (assuming $n_e > n_o$), we need to turn $(n_e - n_o)/2$ even vertices into odd vertices so that \hat{T} is balanced. What we can do is change even leaves to odd vertices since leaves are easier to manage and Lemma 3 assures that there are at least $n_e + n_o - 1$ even leaves. The simplest way of turning an even leaf l_1 to an odd vertex is to delete the edge connecting l_1 and add a new edge between l_1 and its grandparent (or sibling) provided that the degree of its grandparent (or sibling) is not 3. It is obvious that the resulting tree is a more balanced binary tree and the edge between l_1 and its parent is dilated to a distance of 2. The following three-step procedure is how we transform T to the desired \hat{T} by utilizing siblings and grandparents of even leaves.

1. If any two even leaves l_1 and l_2 share a parent (i.e. l_1 and l_2 are siblings), move l_2 so that l_2 becomes the child of l_1 (Figure 3-a). Repeat until no even leaves

- share a parent or the tree is balanced.
2. If any two even leaves l_1 and l_2 share a grandparent, rearrange the relative positions of the vertices as shown in Figure 3-b. Repeat until no even leaves share a grandparent or the tree is balanced.
 3. If an even leaf l_1 has a grandparent with degree less than 3, move l_1 so that l_1 becomes the child of its grandparent (Figure 3-c). Repeat until the tree is balanced.

To show that this procedure works, we need the following lemma which states the relation between the number of even leaves and the number of degree 3 even vertices in a binary tree.

LEMMA 8. *Given a binary tree T with n_e even vertices, n_o odd vertices, ($n_e \geq n_o$), let x be the number of even leaves and y be the number of degree 3 even vertices in T . Then $x - y = n_e - n_o + 1$.*

Proof. The sum of the degree of the even vertices is $degree(even) = x + 2(n_e - x - y) + 3y = 2n_e - x + y$. Since $degree(even)$ equals the number of edges in the tree, $n_e + n_o - 1 = 2n_e - x + y$ and $x - y = n_e - n_o + 1$. \square

LEMMA 9. *An arbitrary binary tree T can be transformed into a balanced binary tree \hat{T} such that $|T| = |\hat{T}|$ and adjacent vertices in T are at most distance 2 in \hat{T} .*

Proof. Using the three-step procedure outlined above, we will show that T can always be transformed into the desired balanced binary tree \hat{T} . Step 1 is trivial; the resulting tree T' of step 1 is obviously more balanced than T and adjacent vertices in T are at most distance 2 in T' . In step 2, since even leaves that are siblings have been moved in step 1, the lowest common ancestor of l_1 and l_2 must be their common

grandparent. Thus the situation is as shown in Figure 3-b. It is easy to check that adjacent edges in T are at most distance 2 in the resulting tree T'' . Note that the subtrees X and Y in Figure 3-b stay adjacent to their parents after the transformation. Thus distance 2 is preserved even if the roots of X and Y are grandparents of some even leaves involved in other transformations. After step 2, we are assured that no even leaves share grandparents. By Lemma 8 there are at least $n_e - n_o + 1$ even leaves in T'' whose grandparents are not degree 3. Thus we can always move $(n_e - n_o)/2$ even leaves to their grandparents to balance T'' . Hence the resulting tree \hat{T} of step 3 satisfies the requirements. \square

With Lemma 9 we can now show that good embeddings for balanced binary trees can be extended to good embeddings for arbitrary binary trees with only slight penalty on dilation.

LEMMA 10. *If all balanced binary trees can be embedded into hypercubes with expansion e and dilation d , then all binary trees can be embedded into hypercubes with expansion e and dilation $d + 1$.*

Proof. Simply transform T into \hat{T} using the three-step procedure. Since \hat{T} is balanced, \hat{T} can be embedded into hypercubes with expansion e and dilation d . The embedding used for \hat{T} is an embedding for T with expansion e and dilation $d + 1$. \square

The results of this section imply that Havel's conjecture is at least as strong as another well-known conjecture attributed to Nebeský [11]: every binary tree can be embedded with expansion 1 and dilation 2 and (using a different embedding) with dilation 1 and expansion 2. If Havel's conjecture is true, Lemma 7 and the remarks after it imply that all binary trees can be embedded into hypercubes with expansion 2

and dilation 1. Also, if Havel’s conjecture is true, Lemma 10 implies that all binary trees can be embedded into hypercubes with expansion 1 and dilation 2. These results are the best possible (assuming Havel’s conjecture holds), since complete binary trees require expansion 2 for a dilation 1 embedding or dilation 2 for an expansion 1 embedding.

6. Conclusions. This paper has shown that all binary trees can be embedded into hypercubes with expansion 1 and average dilation no more than 2. This is an improvement over known results given in [4] and [10] when average dilation is of interest. Our results also extend to arbitrary d -ary trees: all d -ary trees can be embedded into hypercubes with expansion 1 and average dilation no more than $O(\log d)$ — see [5].

In addition, we showed that good embeddings of balanced binary trees can be extended to arbitrary binary trees. These results imply that Havel’s conjecture is at least as strong as another well-known conjecture, namely that every binary tree can be embedded with expansion 1 and dilation 2 or with dilation 1 and expansion 2. We further conjectured that all binary trees can be embedded into hypercubes with expansion 1 and average dilation approaching 1. So far known results on embedding binary trees into hypercubes such as [4], [10], [12], and the strategy presented in this paper have not been able to match the lower bounds. We believe that further studies on the embeddings of balanced binary trees are the key to improving known results.

Our experiments with general purpose heuristics such as greedy [7] or local search [6] show that they do a good job of embedding binary trees. In fact they do better on average (on a 7-cube) than the preorder-Gray-code embedding. However, the preorder-Gray-code embedding has the advantage of being simple, efficient, and analyzable. To further improve the upper bound for average dilation, we believe that a much larger

number of edges must be embedded in dilation 1, as the preorder-Gray-code strategy only embeds half of the edges of the tree in dilation 1. It is not known whether the problem of deciding whether an arbitrary binary tree is a subgraph of a k -cube is NP-hard or can be solved in polynomial time. The status of this problem appears to be closely related to Havel's conjecture.

REFERENCES

- [1] F. AFRATI, C. PAPADIMITRIOU, AND G. PAPAGEORGIOU, *The complexity of cubical graphs*, Information and Control, 66 (1985), pp. 53 – 60.
- [2] S. BHATT AND J.-Y. CAI, *Take a walk, grow a tree*, in Proceedings 29th Annual Symposium on Foundations of Computer Science, 1988, pp. 469 – 478.
- [3] S. BHATT AND C. LEISERSON, *How to assemble tree machines*, in Advances in Computing Research 2, F. Preparata, ed., JAI Press, 1984, pp. 95 – 114.
- [4] S. N. BHATT, F. R. K. CHUNG, F. T. LEIGHTON, AND A. L. ROSENBERG, *Efficient embeddings of trees in hypercubes*, SIAM Journal on Computing, 21 (1992), pp. 151–162.
- [5] W.-K. CHEN, *Theoretical and Experimental Approaches for the Hypercube Embedding Problem*, PhD thesis, North Carolina State University, 1991.
- [6] W.-K. CHEN AND M. STALLMANN, *Local search variants for hypercube embedding*, in Proceedings Fifth Distributed Memory Computing Conference, 1990, pp. 1375 – 1383.
- [7] W.-K. CHEN, M. STALLMANN, AND E. GEHRINGER, *Hypercube embedding heuristics: An evaluation*, International Journal on Parallel Programming, 18 (1989), pp. 505–549.
- [8] I. HAVEL, *On hamiltonian circuits and spanning trees of hypercubes*, Časopis pro Pěstování Matematiky, 109 (1984), pp. 135 – 152.
- [9] I. HAVEL AND P. LIEBL, *One-legged caterpillars span hypercubes*, Journal of Graph Theory, 10 (1986), pp. 69 – 77.
- [10] B. MONIEN AND I. SUDBOROUGH, *Simulating binary trees on hypercubes*, in VLSI Algorithms and Architectures: 3rd Aegean Workshop on Computing, Lecture Notes in Computer Science

319, Springer Verlag, 1988, pp. 170 – 180.

- [11] L. NEBESKÝ, *On cubes and dichotomic trees*, Časopis pro Pěstování Matematiky, 99 (1974), pp. 164 – 167.
- [12] A. WAGNER, *Embedding arbitrary binary trees in a hypercube*, Journal of Parallel and Distributed Computing, 7 (1989), pp. 503–520.
- [13] A. WU, *Embedding of tree networks into hypercubes*, Journal of Parallel and Distributed Computing, 2 (1985), pp. 238 – 249.

FIG. 1. *The algorithm for preorder-Gray-code embedding*

FIG. 2. *Preorder-Gray-code embedding*

FIG. 3. *Increasing odd vertices and decreasing even vertices*

Biographies of authors.

Woei-Kae Chen received the diploma in Electronic Engineering from National Taipei Institute of Technology, Republic of China, in 1984, the M.S. and Ph.D. degree in Computer Engineering from North Carolina State University in 1988 and 1991. Since the fall of 1991 he has been an associate professor of the department of Electronic Engineering, National Taipei Institute of Technology, Republic of China. Dr. Chen's research interests include parallel and distributed computing, graph algorithms, and combinatorial optimization.

Matthias Stallmann received the B.A. degree in Mathematics and Computer Science from Yale University in 1974, the M.S. in Computer Science from Yale in 1978, and the Ph.D. in Computer Science from the University of Colorado at Boulder in 1982. From January, 1983 to June, 1984 he was Visiting Assistant Professor in the Department of Mathematics and Computer Science at the University of Denver. Since the fall of 1984 he has been on the Computer Science faculty at North Carolina State University. Dr. Stallmann's research interests include graph algorithms, combinatorial optimization, and VLSI routing. He is a member of the Association for Computing Machinery, the Society for Industrial and Applied Mathematics, the European Association for Theoretical Computer Science, and Omega Rho.

```

procedure pgc( $v$ ) is    /* preorder-Gray-code embedding */
begin
     $f(v) := g(i)$ ;        /*  $i$  is a global variable */
     $i := i + 1$ ;
    if  $v$  is not a leaf then
        Let the children of  $v$  be  $L(v)$  and  $R(v)$  with
             $s(L(v)) \leq s(R(v))$ ;
        pgc( $L(v)$ );
        pgc( $R(v)$ );        /* if it exists */
    endif
end

```

FIG. 1. The algorithm for preorder-Gray-code embedding

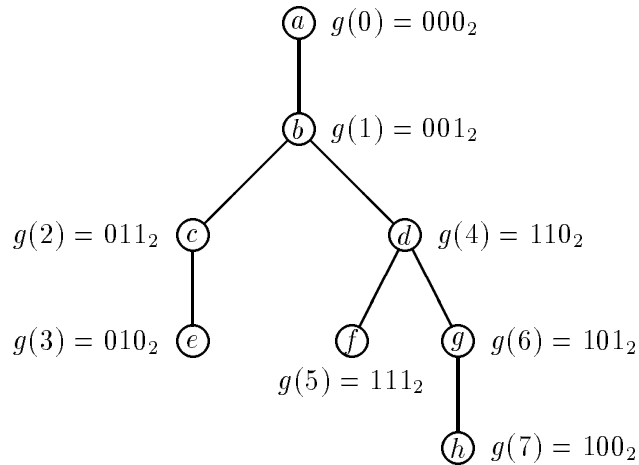
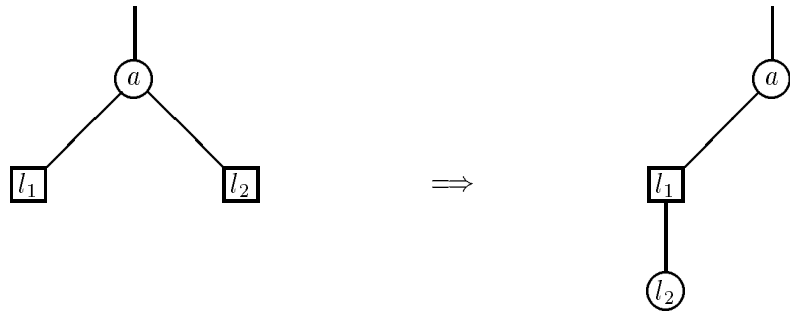
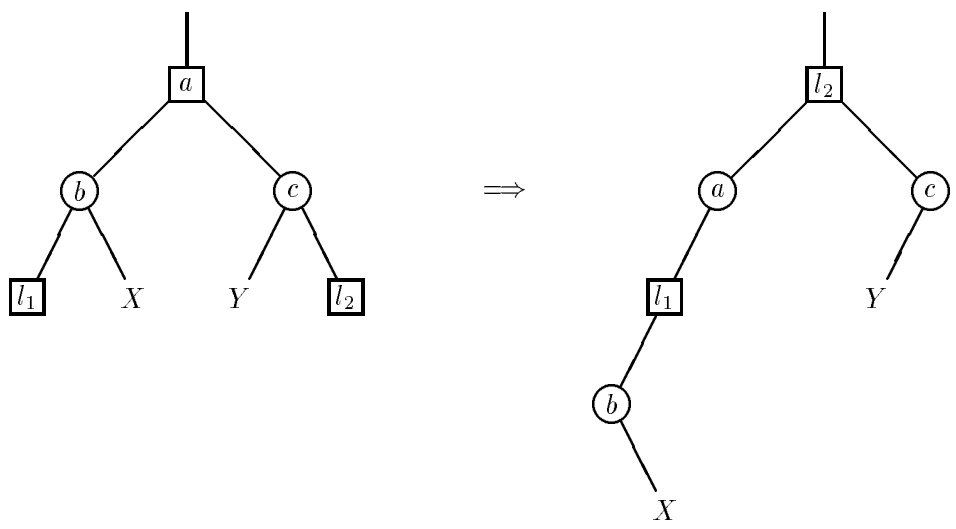


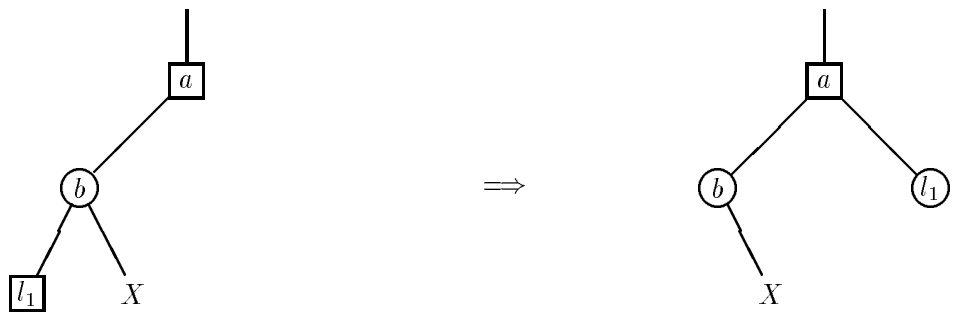
FIG. 2. Preorder-Gray-code embedding



(a)



(b)



(c)

Vertices enclosed by \square are even vertices, \circ are odd vertices. X and Y are subtrees.

FIG. 3. Increasing odd vertices and decreasing even vertices