Optimal Independent Spanning Trees on Hypercubes

Shyue-Ming Tang and Yue-Li Wang

All correspondence should be addressed to Professor Yue-Li Wang, Department of Information Management, National Taiwan University of Science and Technology, 43, Section 4, Kee-Lung Road, Taipei, Taiwan, Republic of China (e-mail: ccdir@mail.ntust.edu.tw).

This work was supported by the National Science Council, Republic of China, under Contract 91-2213-E-011-043.

Abstract

Two spanning trees rooted at some vertex r in a graph G are said to be *independent* if for each vertex v of G, $v \neq r$, the paths from r to v in two trees are vertex-disjoint. A set of spanning trees of G is said to be independent if they are pairwise independent. A set of independent spanning trees is *optimal* if the average path length of the trees is minimum. Any k-dimension hypercube has k independent spanning trees rooted at an arbitrary vertex. In this paper, a linear time algorithm is proposed to construct k optimal independent spanning trees on a k-dimension hypercube.

Keywords: independent spanning trees, hypercube, fault-tolerant broadcasting, product graph.

1. Introduction

A k-dimension *hypercube*, denoted by Q_k, is a graph $G = (V, E)$ with $V = \{0,1,2,..., 2^k-1\}$ and $E = \{(u,v) \mid v \oplus u = 2^i, 0 \le i \le k-1\}$, where $\oplus$ denotes a k-bit exclusive or operation. Thus, if k is a positive integer, Q_k is both k-connected and k-regular. Meanwhile, the hypercube is a well-known class of graphs which may be described in terms of product operation, i.e., $Q_k = Q_{k-1} \times K_2$, and $Q_1 = K_2$ is a complete graph with two vertices [4]. For an example, Q_4 is shown in Figure 1.

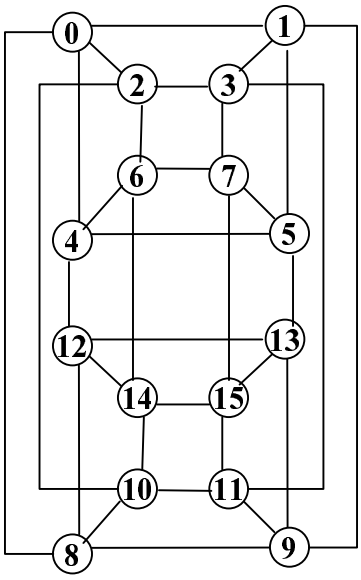

Figure 1 An example 4-dimension hypercube Q_4 .

Hypercubes (or hypercube networks) are important due to their simple structure and suitability for developing algorithms [1, 2, 6, 12, 13, 14, 17, 21]. There are commercially available parallel computers, such as nCUBE [18], CM-5 [8,9], and iPSC [19], which are equipped with the hypercube multiprocessors architecture. Therefore, it is valuable to investigate the communication problems on hypercubes.

A set of paths connecting two vertices in a graph is said to be *internally disjoint* if any pair of paths in the set have no common vertex except the two end

vertices. Considering a graph $G=(V,E)$, a tree T is called a *spanning tree* of G if T is a subgraph of G and T contains all vertices in V. Two spanning trees of G are said to be *independent* if they are rooted at the same vertex, say r, and for each vertex $v \neq r$, the two paths from r to v, one path in each tree, are internally disjoint (or called vertex-disjoint). A set of spanning trees of G is said to be independent if they are pairwise independent. For example, four independent spanning trees of the hypercube Q_4 are shown in Figure 2. For each vertex $v \in \{1,2,...,15\}$, four paths from 0 to v, one path in each tree, are internally disjoint.

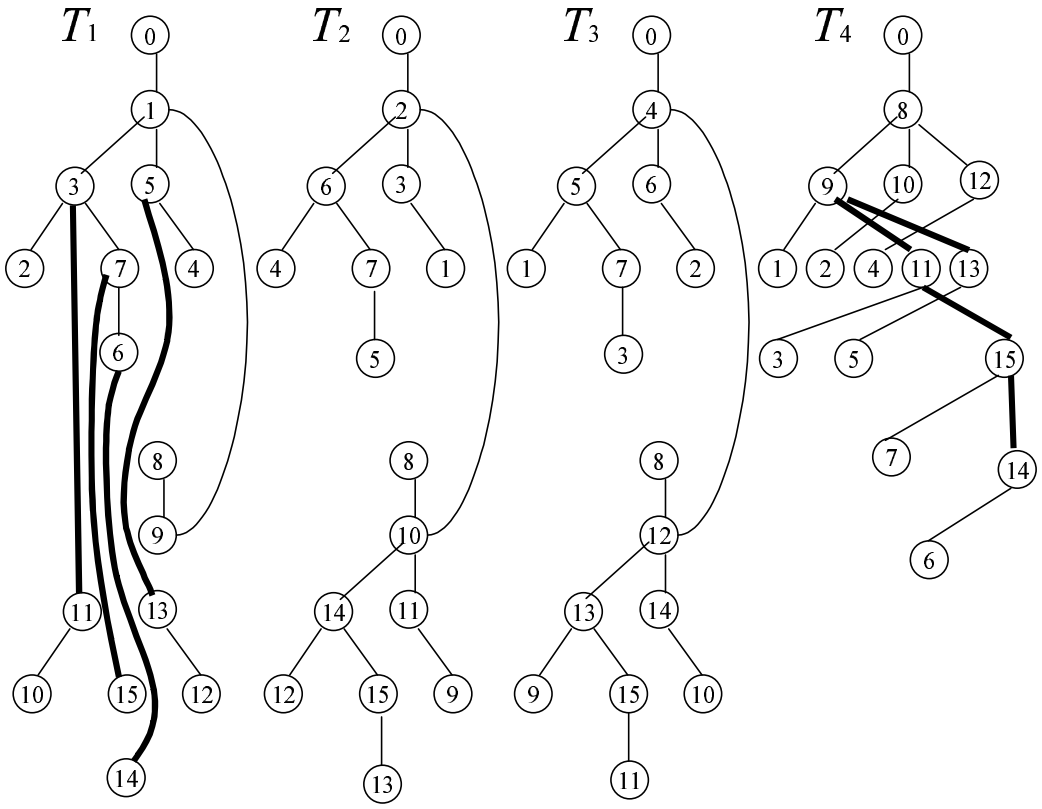

Figure 2 Four independent spanning trees on Q_4 .

The study on independent spanning trees has applications in fault-tolerant protocols for distributed computing networks. For example, broadcasting in a network is sending a message from a given node to all other nodes in the network. A fault-tolerant broadcasting protocol can be designed by means of

independent spanning trees [3,11]. The fault-tolerance is achieved by sending k copies of the message along k independent spanning trees rooted at the source node. If the source node is faultless, this scheme can tolerate up to k–1 faulty nodes.

In [11], Itai and Rodeh gave a linear time algorithm for finding two independent spanning trees in a biconnected graph. In [5], Cheriyan and Maheshwari showed that, for any 3-connected graph G and for any vertex r of G, three independent spanning trees rooted at r can be found in $O(|V||E|)$ time. In [23] and [15], the authors conjectured that any k-connected graph has k independent spanning trees rooted at an arbitrary vertex r. In [10], Huck has proved that the conjecture is true for the class of planar graphs. The conjecture is still open for arbitrary k-connected graphs with $k > 3$.

In [20], Obokata et al. mentioned that an k-dimension hypercube is an k-channel graph and has k independent spanning trees rooted at any vertex. According to their scheme, an k-dimension hypercube Q_k can be viewed as the product graph of Q_{k-1} and K_2, and k independent spanning trees on Q_k can be constructed recursively from k–1 independent spanning trees on Q_{k-1}. Four independent spanning trees on Q_4 shown in Figure 2 are constructed by Obokata's algorithm that will be introduced in Section 3. However, the tree set constructed by Obokata's algorithm is not optimal in terms of average path length when $k > 3$. To make up the shortcoming, in this paper we propose an algorithm to construct k optimal independent spanning trees on a k-dimension hypercube.

The remaining part of this paper is organized as follows. In Section 2, we introduce some notation and define the optimal independent spanning trees rooted at a vertex in a given graph. In Section 3, we propose an algorithm for constructing k optimal independent spanning trees on a k-dimension hypercube. In Section 4, we give the correctness proof for the algorithm. The last section contains our concluding remarks.

2. Definition of Optimal Independent Spanning Trees

Let $d_G(u,v)$ denote the *distance*, i.e., the number of edges, between vertices u and v in G. The *height* of a tree T rooted at vertex r is the maximum distance of the paths from r to any other vertices in T. In [16], the *path length* of a tree is defined as the sum of the distance from every vertex to the root of the tree. The path length is a natural concept when we analyze the search cost of a tree. Let G be a k-connected graph and $S = \{T_1, T_2, ..., T_k\}$ be a set of independent spanning trees rooted at r in G and $D(v)$ denote the *average distance* from v to r with respect to S, i.e., $D(v) = \sum_{i=1}^{k} d_{Ti}(r,v) / k$. Then, we define the *average path length* of S as the summation of $D(v)$, for all $v \in V$ and $v \neq r$. That is, the average path length of S is equal to $\sum_{v \in V \setminus \{r\}} D(v)$, or $\sum_{i=1}^{k} \sum_{v \in V \setminus \{r\}} d_{Ti}(r,v) / k$. A set S of independent spanning trees is defined to be *optimal* if the average path length of S is minimum. For example, the average path length of the tree set in Figure 2 is (46+46+46+50)/4 = 47, while the average path length of the tree set in Figure 3 is (46+46+46+46)/4 = 46. We are going to explain that a set of independent spanning trees in Figure 3 is optimal since its average path length is minimum.

We define child(v,i), parent(v,i) as the *children set* and the *parent set*, respectively, of vertex v in T_i of a tree set S. The *ancestor set* of a vertex v in T_i, denoted by ancestor(v,i), is the set of vertices in the path from r to parent(v,i) in T_i. The *descendant set* of a vertex v in T_i, denoted by descendant(v,i), is the set of all vertices except v in the subtree rooted at v in T_i. The *neighborhood* of a vertex v, denoted by $N(v)$, is the set of all vertices which are adjacent with v in a graph.

A k-dimension hypercube is both k-connected and k-regular. In [22], the authors proposed a *parent exchange* scheme to reduce the height of some independent spanning tree in a k-connected and k-regular graph. A *parent*

exchange of vertex v with respect to a set of k independent spanning trees is to change the parent of v from vertex parent(v,i) to vertex parent(v,π_i), where (π_1 π_2 ... π_k) is a permutation of (1 2 ... k). For example, the tree set in Figure 3 results from a parent exchange (4132) of vertex 14 with respect to the tree set in Figure 2. That is, the parent of vertex 14 in T_1 is changed to parent(14,π_1) = parent(14,4) = {15}; the parent of vertex 14 in T_2 is changed to parent(14,π_2) = parent(14,1) = {6}; the parent of vertex 14 in T_3 remains unchanged; and the parent of vertex 14 in T_4 is changed to parent(6,π_4) = parent(14,2) = {10}.

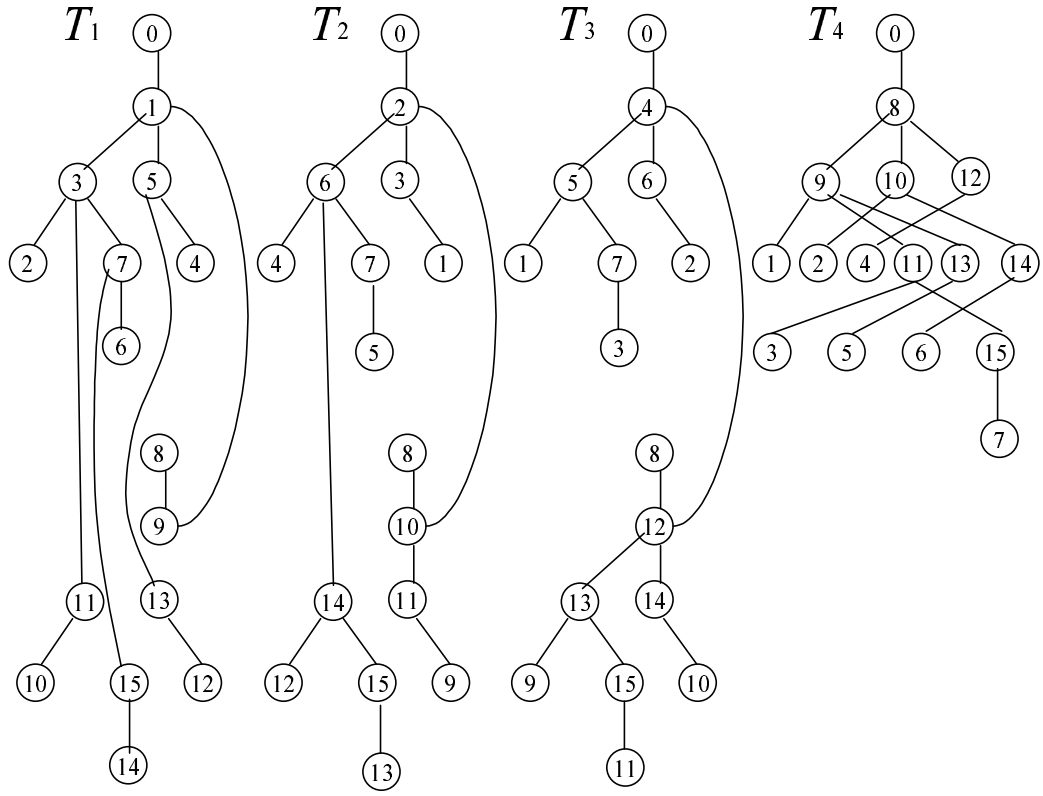

Figure 3 Four optimal independent spanning trees on Q_4 .

A parent exchange may not be beneficial to the height of a set of independent spanning trees. Let {T_1*, T_2*,... , T_k*} denote the tree set after a parent exchange. We define the *benefit* of the parent exchange on some vertex v for T_i as

$$\text{benefit}(v, i) = (|\text{descendant}(v,i)| + 1)\ (x_i - y_i),$$

where x_i and y_i denote the distance from r to v in T_i and T_i* , respectively. A

parent exchange affects not only the distance from r to v but also the distance from r to all descendants of v in T_i. Thus, we have to multiple the distance change with the number of vertices in the subtree rooted at v in T_i.

The *total benefit* of a parent exchange on vertex v is the summation of the benefit with respect to the tree set , i.e.,

$$\sum_{i=1}^{k} \text{benefit}(v, i).$$

A parent exchange is *beneficial* if the total benefit of the exchange is positive. For example, the total benefit of the parent exchange (4132) on vertex 14 with respect to the tree set in Figure 2 is benefit(14,4) = (| descendant(14,4) | +1) (5–3) = $2 \times 2 = 4$.

Based on this definition of a set of optimal independent spanning trees, we have the following lemma.

Lemma 1. *Given a k-connected, k-regular graph G and a set S of independent spanning trees on G, if there is no beneficial exchange in S, then S is optimal.*

Proof. Suppose the tree set S is not optimal. Let $S^* = \{T_1^*, T_2^*, \ldots, T_k^*\}$ be a set of optimal independent spanning trees rooted at the same vertex. Since $\Sigma_{v \in V \setminus \{r\}} D(v)$ is greater than $\Sigma_{v \in V \setminus \{r\}} D^*(v)$, there exists at least one vertex u such that $D(u)$ is greater than $D^*(u)$. It turns out that there exists a beneficial exchange on u with respect to S such that $\Sigma_{i=1}^{k}$benefit(u,i) is positive. This contradicts that no beneficial exchange exists in S. **Q.E.D.**

No beneficial exchange implies a set of optimal independent spanning trees. However, to determine whether there exists a beneficial exchange in a tree set is not easy. The following lemma provides a more efficient method to identify a set of optimal independent spanning trees.

Lemma 2. *Given a k-connected, k-regular graph G and a set S of independent spanning trees rooted at r in G. Let v be a vertex in G,* $v \notin \{r\} \cup N(r)$, *and* $u \in N(v)$, *if* $|d_T(r,u) - d_T(r,v)| \leq 1$ *for every* $T \in S$, *then S is optimal.*

Proof. There is no beneficial exchange for root vertex r and vertices in $N(r)$ since any parent exchange breaks the independency of S, and thus, is not feasible. For vertex $v \notin \{r\} \cup N(r)$ and $u \in N(v)$, since $|d_T(r,u) - d_T(r,v)| \leq 1$ for a tree $T \in S$, the distance from r to u in T is the same as (i) the distance from r to the parent of v, (ii) the distance from r to v, or (iii) the distance from r to one child of v. In all of these cases, there is no benefit for v to change its parent vertex in T. If this condition holds for every tree in S, then there is no beneficial exchange on v with respect to the tree set S. Since for tree set S there is no beneficial exchange on every vertex in G, by Lemma 1, S is optimal. **Q.E.D.**

In Figure 3, for every vertex v, $v \notin \{0,1,2,4,8\}$, if u is a neighbor of v in Q_4, then u is either in the parent layer or in the children layer of v in T_i (i = 1,2,3,4). That is, $|d_{Ti}(r,v) - d_{Ti}(r,u)|$ is always equal to one. It turns out that the set of independent spanning trees is optimal.

3. Construction of Optimal Independent Spanning Trees

Hypercubes are vertex-symmetric [7]. Without loss of generality, we simply consider independent spanning trees rooted at vertex 0 of a hypercube. Before describing our algorithm, we rewrite the algorithm proposed by Obokata et al. [20] as follows.

Algorithm *IST_Hypercube*(k)
Input: k.

Output: A set of k independent spanning trees on Q_k .

Method:

Step 1. **If** k is equal to 2, **then** return path <0,1,3,2> and path <0,2,3,1> as T_1 and T_2 , respectively.

Step 2. **Call** *IST_Hypercube* (k–1).

Step 3. Let T'_1, T'_2, ..., T'_{k-1} be the k–1 independent spanning trees on Q_{k-1} . Construct T''_1, T''_2, ..., T''_{k-1} by adding 2^{k-1} to the value of each vertex in T'_1, T'_2, ..., T'_{k-1} .

Step 4. (Construction of T_1, T_2, ..., T_{k-1})

For i = 1 **to** k–1 **do**

Construct T_i by connecting the only child of the root in T'_i (i.e., vertex 2^{i-1}) with the corresponding vertex in T''_i (i.e., vertex $2^{i-1} + 2^{k-1}$).

Step 5. (Construction of T_k)

Substep 5.1 (Create the only child of the root)

Connect vertex 0 with vertex 2^{k-1} .

Substep 5.2 (Create k–1 grandchildren of the root)

For i = 0 **to** k–2 **do**

Connect vertex 2^{k-1} with vertex $2^{k-1}+2^i$.

Substep 5.3 (Create 2^{k-1}–1 leaves)

For every vertex $v < 2^{k-1}$ and $v \neq 0$ **do**

Connect vertex v with vertex $v+2^{k-1}$.

Substep 5.4 (Create 2^{k-1}–k edges in T_k by transforming T_1)

For every vertex $v \notin \{0,1,2,...,2^{k-1},2^{k-1}+1,2^{k-1}+2^1,...,2^{k-1}+2^{k-2}\}$ **do**

Set parent(v,k) = parent(v,1).

Set parent(v,1) = v–2^{k-1} .

enddo

End of Algorithm *IST_Hypercube*

Based on the algorithm, k independent spanning trees on Q_k are recursively constructed by combining k–1 pairs of identical spanning trees of Q_{k-1} , and adding one more spanning tree. Let $f(n)$ be the running time of Algorithm IST_Hypercube, where n is the number of vertices in Q_k , i.e., $n = 2^k$, Then, we have a recurrence equation that bounds $f(n)$:

$$f(n) = 2\,f(n-1) + cn,$$

where c is a constant. By induction on n, we have $f(n)$ is O(n log$n$), or O(kn). Thus, Algorithm IST_Hypercube takes linear time to construct k independent spanning trees.

Theorem 3. [obo96] *Algorithm IST_Hypercube correctly constructs k independent spanning trees on Q_k in $O(kn)$ time, where $n = 2^k$.*

Four independent spanning trees of Q_4 shown in Figure 2 are constructed by using Algorithm IST_Hypercube. Notice that four bold-line edges in T_1 result from the construction of four corresponding bold-line edges in T_4 . In a k-connected, k-regular graph, two spanning trees rooted at r are not independent if any vertex (not r) has the same parent in two trees. Based on Algorithm IST_Hypercube there are $2^{k-1}-k$ vertices in T_1 that must change their original parents since everyone of them has the same parent in T_k .

By Lemma 1, we know that the tree set in Figure 2 is not optimal since there exists a beneficial parent exchange on vertex 14, i.e., (4131). Furthermore, the height of T_4 can be reduced from 6 to 5 by doing the parent exchange. For reducing the height of T_k , we modify Step 5 of algorithm IST_Hypercube and form a new algorithm as follows.

Algorithm *OIST_Hypercube*(k)

Input: k.

Output: k optimal independent spanning trees on Q_k .

Method:

Step 1. **If** k is equal to 2, **then** return path <0,1,3,2> and path <0,2,3,1> as T_1 and T_2 , respectively.

Step 2. **Call** *OIST_Hypercube*($k-1$).

Step 3. Construct T_1, T_2, ..., T_{k-1} by using the same method in Steps 3 and 4 of Algorithm IST_Hypercube.

Step 4. (Construction of T_k)

Substep 4.1 Create the only child of the root.

Substep 4.2 Create $k-1$ grandchildren of the root.

Substep 4.3 Create $2^{k-1}-1$ leaves.

Substep 4.4 (Create $2^{k-1}-k$ edges in T_k by transforming T_x)

For every vertex $v \notin \{0,1,2,...,2^{k-1},2^{k-1}+1,2^{k-1}+2^1,...,2^{k-1}+2^{k-2}\}$

do

Let T_x be a tree ($1 \le x \le k-1$) in which

$d_{Tx}(0,v)$ is minimum among the $k-1$ trees.

Set parent(v,k) = parent(v,x).

Set parent(v,x) = $v-2^{k-1}$.

enddo

End of Algorithm *OIST_Hypercube*

In Substep 4.4, we choose a T_x , instead of T_1 , as a transformed tree. For example, see Figure 3. We choose T_2 as the transformed tree of vertex 14 since min $\{d_{T_1}(0,14), d_{T_2}(0,14), d_{T_3}(0,14)\}$ = {5, 3, 3} = 3 and x = 2 or 3. Then, we connect vertex 14 to vertex 10 (i.e., parent(14,2)) in T_k and set parent(14,2) to 6 (=14−8). By maintaining an information of minimum distance for a vertex v, we can determine the tree T_x in constant time. Thus, Step 4 takes O(n) time and Algorithm OIST_Hypercube totally takes O(kn) time to construct k independent spanning trees.

4. Correctness

In this section, we shall prove that Algorithm OIST_Hypercube can construct k optimal independent spanning trees on Q_k . Let T_1, T_2, ..., T_k be the output of Algorithm OIST_Hypercube. We have the following lemmas.

Lemma 4. T_1, T_2, ..., *and* T_k *are spanning trees of* Q_k .

Proof. Since T_1, T_2, ..., and T_{k-1} are obtained by using the same method of Algorithm IST_Hypercube, and the transformation step does not affect the property of a spanning tree, T_1, T_2, ..., and T_{k-1} are spanning trees of Q_k . As for T_k , it is obvious that 2^k-1 edges are created in Step 4 and Substeps 4.1, 4.2, and 4.3 do not create cycle. We only have to prove that no cycle is formed

among the edges created by Substep 4.4 of Algorithm OIST_Hypercube.

Let (a,b) be an edge created by Substep 4.4 with b = parent(a,k). It means that there is a tree T_x , $1 \le x \le k-1$, in which $d_{Tx}(0,a)$ is minimum among the $k-1$ trees and $d_{Tx}(0,a) > d_{Tx}(0,b)$. Suppose that there exist a cycle <a,b,c,..., a> in T_x . Then, we have $d_{Tx}(0,a) > d_{Tx}(0,b) \ge d_{Ty}(0,b) > d_{Ty}(0,c) \ge d_{Tz}(0,c) >... \ge d_{Tw}(0,a)$. Since there exists a tree T_w , $1 \le w \le k-1$, and $d_{Tw}(0,a) < d_{Tx}(0,a)$, T_x is not the transformed tree of vertex a. It is a contradiction. Thus, there is no cycle is formed in T_k . **Q.E.D.**

For proving that T_1, T_2, ..., and T_k are pairwise independent, we make use of the proposition : "T_i and T_j $(i \ne j)$ are independent if and only if for every vertex v in Q_k , $v \ne 0$, *ancestor*$(v,i) \cap$ *ancestor*$(v,j) = \{0\}$". For convenience of explanation, we divide Q_k into subgraphs A and B which are induced by vertex sets $\{0,1,2,..., 2^{k-1}-1\}$ and $\{2^{k-1}, 2^{k-1}+1, 2^{k-1}+2, ..., 2^k-1\}$, respectively. Then, T'_i denotes a spanning tree on A, while T''_i denotes a spanning tree on B. Note that both A and B are Q_{k-1} .

Lemma 5. T_1, T_2, ..., *and* T_{k-1} *are mutually independent.*

Proof. T_i $(1 \le i \le k-1)$ is constructed by combining T'_i with T''_i and doing transformation on some vertices. For every vertex v in A, $v \ne 0$, ancestor$(v,i) \cap$ ancestor$(v,j) = \{0\}$ since the two path from v to 0 in T'_i and T'_j $(1 \le j \le k-1)$ are internally disjoint. For every vertex v in B, $v \ne 2^{k-1}$, ancestor$(v,i) \cap$ ancestor(v,j) = {parent(v,i), ..., a, c, ..., child$(0,i)$, 0} $\cap$ {parent(v,j), ..., b, d, ..., child$(0,j)$, 0} = {0} since four vertex sets {parent(v,i), ..., a}, {c, ..., child$(0,i)$}, {parent(v,j), ..., b}, and {d, ..., child$(0,j)$} are mutually disjoint. See Figure 4. In case $v = 2^{k-1}$, ancestor$(v,i) \cap$ ancestor$(v,j) = \{2^{k-1}+2^{i-1}, 2^{i-1}, 0\} \cap \{2^{k-1}+2^{j-1}, 2^{j-1}, 0\} = \{0\}$.

Q.E.D.

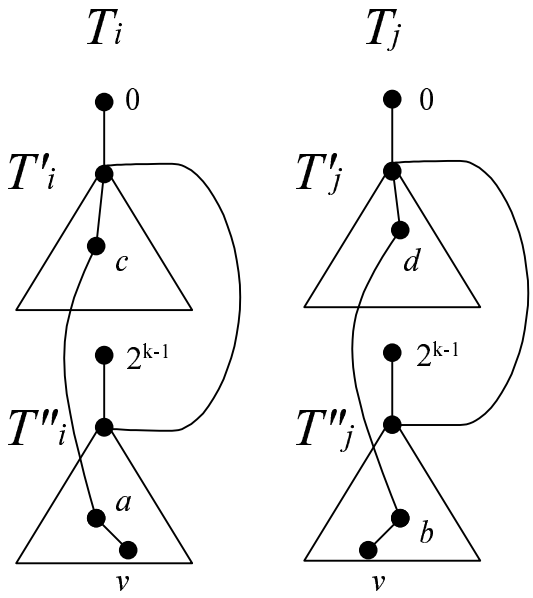

Figure 4 Two paths from v to 0 in T_i and T_j $(1 \le i, j \le k-1)$.

Lemma 6. For $1 \le i \le k-1$, T_k *and* T_i *are independent*.

Proof. Based on Algorithm OIST_Hypercube, if one vertex v is the leaf nodes of T_k , then $v \in A$, else $v \in B \cup \{0\}$. For every vertex v in A, $v \neq 0$, ancestor(v,i) $\cap$ ancestor(v,k) = {parent(v,i), ..., child(0,i), 0} $\cap$ {parent(v,k), ..., child(0,k), 0} = {0} since {parent(v,i), ..., child(0,i)} $\subset A$ and {parent(v,k), ..., child(0,k)} $\subset B$. For every vertex v in B, $v \neq 2^{k-1}$, see Figure 5. {parent(v,i), ..., a} $\cap$ {parent(v,k), ..., child(0,k)} = ϕ due to the transformation step of the algorithm. Thus, ancestor(v,i) $\cap$ ancestor(v,k) = {parent(v,i), ..., a, b, ..., child(0,i), 0} $\cap$ {parent(v,k), ..., child(0,k), 0} = {0} since three vertex sets {parent(v,i), ..., a}, {b, ..., child(0,i)}, and {parent(v,k), ..., child(0,k)} are mutually disjoint. In case $v = 2^{k-1}$, ancestor(v,i) $\cap$ ancestor(v,k) = $\{2^{k-1}+2^{i-1}, 2^{i-1}, 0\} \cap \{0\} = \{0\}$. **Q.E.D.**

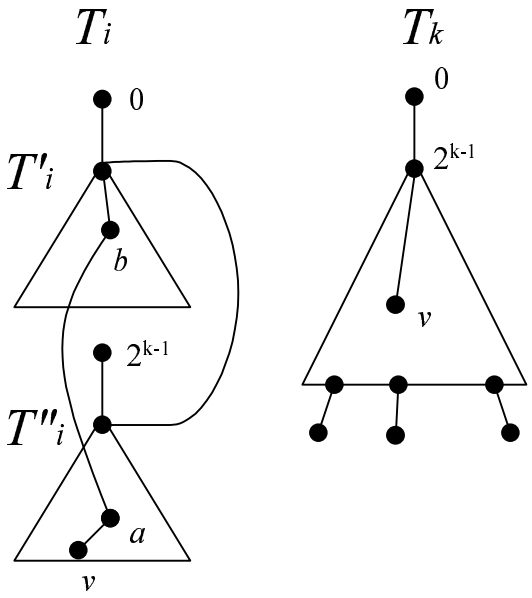

Figure 5 Two paths from v to 0 in T_i and T_k ($1 \le i \le k-1$).

Lemma 7. *Lemma 2 hols in* T_1 , T_2 , ..., T_k .

Proof. We prove this lemma by induction on k. For $k = 2$, Lemma 2 holds in both T_1 and T_2 . Suppose that Lemma 2 holds for $k-1$. Then we should prove that Lemma 2 holds for k.

For $1 \le i \le k-1$, T_i is constructed from T'_i and T''_i . If vertex $v \in A$ and $v \notin \{0\} \cup N(0)$, $N(v)$ consists of $k-1$ vertices in T'_i and one vertex $(v+2^{k-1})$ in T''_i . The optimal property of the $k-1$ neighbors in T'_i retains in T_i . The construction step makes $d_{Ti}(0,v+2^{k-1}) - d_{Ti}(0,v) = 1$. Similarly, if vertex $v \in B$ and $v \ne 2^{k-1}$, $N(v)$ consists of $k-1$ vertices in T''_i and one vertex $(v-2^{k-1})$ in T'_i . The optimal property of the $k-1$ neighbors in T''_i retains in T_i . The construction step makes $d_{Ti}(0,v) - d_{Ti}(0,v-2^{k-1}) = 1$. Since the transformation step do not affect the distance from any vertex to the root, Lemma 2 holds in T_i .

In T_k , if $v \in B$ and $v \ne 2^{k-1}$, then there exists a T_x ($1 \le x \le k-1$) such that $d_{Tk}(0,v) = d_{Tx}(0,v)$ and the distance is minimum among the tree set. If $v \in A$ and $v \ne 0$, then v is a leaf node and its parent is set to $v+2^{k-1}$. It turns out that

$|d_{Tk}(0,u) - d_{Tk}(0,v)| = 1$ for each $u \in N(v)$ since we have proved that $|d_{Ti}(0,u) - d_{Ti}(0,v)| = 1$ ($1 \le i \le k-1$). That is, Lemma 2 also holds in T_k . **Q.E.D.**

We summarize Lemmas 4, 5, 6, and 7 as Theorem 8.

Theorem 8. *Algorithm OIST_Hypercube correctly construct k optimal independent spanning trees on Q_k in $O(kn)$ time, where $n = 2^k$.*

5. Concluding remarks

In this paper, we present a linear time algorithm for constructing k independent spanning trees of a k-dimension hypercube. The height of every spanning tree is $k+1$. This result is optimal since the average path length of the tree set is minimum.

There are many vertex-symmetric graphs, such as star graphs, recursive circulant graphs, and so on. Efficient algorithms to find independent spanning trees for these classes of graphs are valuable. New skills for verifying the independency of spanning trees rooted at a vertex are also eagerly needed when developing these algorithms.

References

[1] B. Abali, F. Ozguner, and A. Bataineh, Balanced Parallel Sort on Hypercube Multiprocessors, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 5, 1993, pp.572-581.

[2] C. Aykanat, F. Ozguner, F. Ercal, and P. Sadayappan, Iterative Algorithms for Solution of Large Sparse Systems of Linear Equations on Hypercubes, *IEEE Transactions on Computers*, Vol. 37, No. 12, 1988, pp.1554-1567.

[3] F. Bao, Y. Igarashi, and S.R. Öhring, Reliable Broadcasting in Product

Networks , *Discrete Applied Mathematics*, Vol. 83, 1998, pp.3-20.

[4] G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory*, McGraw-Hill, Inc., 1993, pp.29-30.

[5] J. Cheriyan, S. N. Maheshwari, "Finding Nonseparating Induced Cycles and Independent Spanning Trees in 3-connected Graphs," *Journal of Algorithms* 9, 1988, pp.507-537.

[6] A. K. Gupta and S. E. Hambrusch, Multiple Network Embeddings into Hypercubes, *Journal of Parallel and Distributed Computing*, Vol. 19, 1993, pp.73-82.

[7] Frank Harary, "Graph Theory," Addison-Wesley, 1968, pp.171-173.

[8] W. D. Hillis. The Connection Machine, MIT Press, 1985.

[9] W. D. Hillis and L. W. Tucker, The CM-5 Connection Machine: A Scalable Supercomputer, *Communications of the ACM*, Vol. 36, No. 11, 1993, pp.30-40.

[10] A. Huck, "Independent Trees in Planar Graphs," *Graphs and Combinatorics* 15, 1999, pp.29-77.

[11] A. Itai and M. Rodeh, The Multi-tree Approach to Reliability in Distributed Networks, in *Proceedings of the 25th Annual IEEE Symposium on Foundation of Computer Science*, 1984, pp.137-147. (Seen also in *Information and Computation*, Vol. 79, 1988, pp.43-59.)

[12] S. L. Johnsson, Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures, *Journal of Parallel and Distributed Computing*, Vol. 4, 1987, pp.133-172.

[13] S. L. Johnsson and C. T. Ho, Optimum Broadcasting and Personalized Communication in Hypercubes, *IEEE Transactions on Computers*, Vol. 38, No. 9, 1989, pp.1249-1268.

[14] J. F. Jenq and S. Sahni, All Pairs Shortest Paths on a Hypercube Multiprocessor, *Proceedings of the International Conference on Parallel Processing*, 1987, pp.713-716.

[15] Samir Khuller and Baruch Schieber, “On Independent Spanning Trees”, *Information Processing Letters* 42, 1992, pp.321-323.

[16] D. E. Knuth, *The Art of Computer Programming*, Vol. 3: Sorting and Searching, Addison-Wesley, 1973, pp.194-198.

[17] F. T. Leighton, Hypercubes and Related Networks, Chapter 3 in *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann Publishers, Inc., 1992.

[18] nCUBE Corporation, nCUBE 2S Programmer's Reference Manual, 1992.

[19] S. F. Nugent, The iPSC/2 Direct-connect Communications Technology, *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications*, Pasadena, 1988, pp.51-60.

[20] K. Obokata, Y. Iwasaki, F. Bao, and Y. Igarashi, “Independent Spanning Trees of Product Graphs,” *Lecture Notes in Computer Science* 1197, 1996, pp.338-351.

[21] Y. Saad and M. H. Schultz, Topological Properties of Hypercube, *IEEE Transactions on Computers*, Vol. 37, No. 7, 1988, pp.867-872.

[22] S. Tang, , Yue-Li Wang, and Jing-Xian Lee, “On the Height of Independent Spanning Trees of A k-connected k-regular Graph,” *Proceedings of National Computer Symposium*, Taipei, 2001, pp.A159-A164.

[23] A. Zehavi, A. Itai, “Three Tree-paths,” *Journal of Graph Theory* 13, 1989, pp.175-188.