How Many Optimal Independent Spanning Trees are there in a Hypercube†

Shyue-Ming Tang
Department of Psychology and Social Work
National Defense University
Taipei City, Taiwan, R.O.C.
tang1119@gmail.com

Abstract
Suppose multiple spanning trees rooted at same vertex \( r \) in a graph. They are mutually independent if for each non-root vertex \( v \), paths from \( v \) to \( r \), one path in each spanning trees, are internally disjoint and no common edge. It has been proved that there exist \( n \) optimal independent spanning trees (OIST for short) rooted at any vertex in the \( n \)-dimensional hypercube (denoted by \( Q_n \)). The optimal requirement is achieved when the distance from \( v \) to the only child of \( r \) is the Hamming distance in every spanning tree. In this paper, the number of OIST solutions in \( Q_n \) is concerned. A routing square is an \( n \) by \( n \) matrix, which can represent an OIST solution, but not every OIST solution has a correspondent routing square. Let \( R_n \) denote the number of routing squares of \( Q_n \) and let \( f(m) \) denote the number of \( m \) 1-bits vertices in \( Q_n \). An upper bound and a lower bound for the number of OIST solutions in \( Q_n \) are proved to be \( \Pi_{m=3}^{n-1} (\frac{m}{m^n}) \) and \( R_n \), respectively, where \( \Psi_n \) is the derangement number of \( m \).

Keywords: hypercubes, internally disjoint paths, Hamming distance, optimal independent spanning trees, fault-tolerant broadcasting, derangement number.

1. Introduction
The \( n \)-dimensional hypercube, denoted by \( Q_n \), is defined as a graph with vertex set \( \{0, 1, 2, \ldots, 2^n - 1\} \) and edge set \( \{(u, v) : \text{dist}(u, v) = 2^i, 1 \leq i \leq n\} \). Hypercubes are an important class of interconnection networks. They are a subclass of bipartite graphs, as well as a subclass of product graphs [8]. Hypercubes are one of the most famous topology for developing network algorithms.

In computer networking, broadcasting is a method of transferring a message to all recipients simultaneously. Broadcasting can be implemented by a spanning tree rooted at the sender. In the past, some research concerned with fault-tolerant broadcasting on hypercubes [2, 5, 6, 9, 10, 11]. In [10], Tang et al. proposed an algorithm based on constructing optimal independent spanning trees (OIST for short). A parallelized version of this algorithm was proposed in [11].

Multiple spanning trees on a graph are called independent if they are rooted at a common vertex \( r \), and for each vertex \( v \neq r \), any two paths from \( v \) to \( r \) (one path in a tree) are internally disjoint and no common edge [4]. In [12], Zehavi and Itai conjecture that, for any vertex \( r \) in a \( k \)-connected graph, there exist \( k \) independent spanning trees (ISTs for short) rooted at \( r \) in the graph. This conjecture has been proved for \( k \)-connected graphs with \( k \leq 4 \) [1, 3, 4].

Hypercubes are vertex-symmetric. Without loss of generality, vertex 0 is assigned to the root of all trees in an IST solution. Further, since \( Q_n \) is \( n \)-regular, the root vertex has exactly one child in the \( n \) IST of \( Q_n \). Thus we can use \( T_i \) to denote the spanning tree where vertex \( 2^i-1 \) is the only child of the root, i.e., \( T_1, T_2, \ldots, T_n \) denote an IST solution. The \( n \) IST become optimal if they satisfy the requirement that the distance from every vertex to the only child of the root is the Hamming distance in every spanning tree. Hereafter, we use \( \Psi_n \) to stand for an OIST solution of \( Q_n \). Figure 1 shows \( Q_4 \) and a \( \Psi_4 \).

![Figure 1 Q_4 and a \( \Psi_4 \)]
disjoint, meanwhile the distance between every vertex to vertex $2^{n-1}$ in $T_i$ is the Hamming distance.

Let $n$ denote the derangement number or subfactorial of $n$, which is defined as the number of permutations of $n$ elements with none at its original position [13]. The following two recurrences for the derangement number were proved by Euler: $!n = (n-1) \cdot (!(n-1) + !(n-2))$ and $!n = n \cdot !(n-1) + (-1)^n$. By this definition, we have $!1 = 0$, $!2 = 1$, $!3 = 2$, $!4 = 9$, $!5 = 44$, and so forth [14]. The derangement number can be invoked to give an upper bound for the number of $\Psi_n$ as discussed in Section 3.

Vertex $2^n-1$ is routed a path to vertex 0 (the root) in every tree. If a vertex $x$ takes jump $j$ to reach its parent $y$, it means $y = x - j$. A routing square is an $n$ by $n$ matrix $[r_{ij}]$, which records the $j$-th jump from vertex $2^n-1$ to vertex 0 (the root) in $T_i$, where $1 \leq i, j \leq n$. For example, the routing square of the $S_4$ in Figure 1 is

$$
\begin{bmatrix}
2 & 4 & 8 & 1 \\
4 & 8 & 1 & 2 \\
8 & 1 & 2 & 4 \\
1 & 2 & 4 & 8
\end{bmatrix}
$$

In Section 4, we will describe the details about routing squares and explain how one routing square can be transformed to a number of $\Psi_n$'s, hence a lower bound for the number of $\Psi_n$ is induced.

The remaining part of this paper is organized as follows. In Section 2, we introduce some important properties about OIST on hypercubes. Section 3 contains the proof of an upper bound for the number of $\Psi_n$. Section 4 gives the proof of a lower bound. The last section is the concluding remarks.

2. PROPERTIES OF OIST ON HYPERCUBES

In this section, we introduce some notations and important properties of $\Psi_n$.

The distance between vertices $x$ and $y$ in tree $T_i$ is denoted by $d(x,y)$, while the Hamming distance between them is denoted by $d_H(x,y)$. The binary representation of vertex $x$ in $Q_n$ is denoted by $x_p s_{n-1} \ldots s_2 s_1$. The parent of vertex $x$ in tree $T_i$ is denoted by $p(x)$. Let $b(x)$ denote the bit where vertices $x$ and $p(x)$ are different in tree $T_i$. That is, if $y = p(x)$ and $b(x) = j$, then $y = x - 2^j$ or $x + 2^j$ when $x_j = 0$ or 0, respectively. Then, we give two necessary conditions for constructing a $\Psi_n$.

**Lemma 2.1.** For every $T_i \in \Psi_n$, if $p(x) \neq r$ and $x_i = 1$, then $b(x) \neq i$, for $1 \leq i \leq n$.

**Proof.** Let $y = p(x)$ and $r = p(z)$. If $x_i = 1$ and $b(x) = i$, then $y_i = 0$. Since $z_i = 1$, $d_H(x,z) > d_H(x,z)$. This contradicts that $T_i \in \Psi_n$ and the lemma follows.

Q.E.D.

**Lemma 2.2.** For every $T_i \in \Psi_n$, if $x \neq r$ and $x_i = 0$, then $b(x) = i$, for $1 \leq i \leq n$.

**Proof.** Suppose to the contrary that $b(x) \neq i$. There must be a $T_j$ with $j \neq i$ such that $b(x) = i$. Let $y = p(x)$ and $r = p(z)$. If $x_i = 0$, then we have $y_i = 1$ in $T_j$. By the property of $T_j$, $z_i = 0$ and $z_i = 1$. This reveals that $d_H(x,z) > d_H(x,z)$. It is a contradiction and the lemma follows.

Q.E.D.

The following lemma which is deducted from Lemmas 2.1 and 2.2 describes another property of $\Psi_n$.

**Lemma 2.3.** For every $T_i \in \Psi_n$, $x$ is a leaf in $T_i$, if and only if $x \neq r$ and $x_i = 0$, for $1 \leq i \leq n$.

**Proof.** Suppose to the contrary that $x_i = 0$ and $x$ is not a leaf in $T_i$. Accordingly, $x$ has a child $y$. Let $x = p(y)$. If $y_i = 1$, then, by Lemma 2.1, $b(y) \neq i$, namely $x_i = 1$. On the other hand, if $y_i = 0$, then, by Lemma 2.2, $b(y) = i$, namely $x_i = 1$. Both cases result in a contradiction to $x_i = 0$. Therefore, $x$ must be a leaf in $T_i$ if $x_i = 0$.

In case of $x_i = 1$, it is obvious that $x$ is not a leaf in $T_i$ when $p(x) = r$. For the case where $p(x) \neq r$, there exists a neighbor of $x$ in $Q_n$, say $y \neq r$, whose $y_i = 0$.

By Lemma 2.2, $b(y) = i$. This implies that $x = p(y)$ and $x$ is not a leaf in $T_i$ when $x_i = 1$. This concludes the lemma.

Q.E.D.

**Corollary 2.4.** For every $T_i \in \Psi_n$, there are exactly $2^{n-1}-1$ leaves in $T_i$, for $1 \leq i \leq n$.

3. AN UPPER BOUND OF THE NUMBER OF $\Psi_n$

The only child of the root in every tree of $\Psi_n$ is distinct, while the parent of a non-root vertex in every tree is also distinct. It turns out that the OIST construction is to determine a bijection from the parent set to the tree set for every non-root vertex. By Lemma 2.2, $p(x)$ is determined when $x_i = 0$. We have to know the number of possible $p(x)$ when $x_i = 1$. And then, we can determine an upper bound for the number of $\Psi_n$.

Based on Lemmas 2.1 and 2.2, we have the following Lemma.

**Lemma 3.1.** For a non-root vertex $x$ in $Q_n$, $x \neq 2^i \cdot (1 \leq i \leq n)$, the number of possible combinations of $b(x)$ in a $\Psi_n$ is $!m$, where $m \geq 2$ is the number of 1-bits of $x$.

**Proof.** Based on Lemma 2.2, $b(x) = i$ is determined if $x_i = 0$. Based on Lemma 2.1, on the other hand, $b(x) \neq i$ if $x_i = 1$. Since $m \geq 2$, there must exist a $j \neq i$ such that $x_j = 1$ and $b(x) = j$. By the definition of the derangement number, the number of possible combinations of $b(x)$ ($i = 1, 2, \ldots, n$) is the derangement number of $m$.

**Corollary 3.2.** For a non-root vertex $x$ in $Q_n$, if the number of 1-bits of $x$ is less than 3, then $p(x)$ ($i = 1, 2, \ldots, n$) is determined.

In case the number of 1-bits in $x$ is great than 2, we call the vertex a parent-pending vertex. That is, $b(x)$ ($i = 1, 2, \ldots, n$) has more than one possible combinations. Further, we can compute the number of parent-pending vertices in $Q_n$ is
\[ \sum_{i=3}^{n} C_i^n = 2^n - n(n+1)/2 - 1. \]

In Figure 1, the solid lines in the OIST represent the determined parent relationship, while the dotted lines represent the parent-pending relationship. In \( Q_n \), vertices 1 (0001), 2 (0010), 3 (0111), 4 (0100), 5 (0101), 6 (0110), 8 (1000), 9 (1001), 10 (1010) and 12 (1100) have determined parents (by Corollary 3.2). Vertices 7 (0111), 11 (1011), 13 (1101) and 14 (1110) have \( \exists = 2 \) possible \( p(x) \) combinations (or bijections from the parent set to the tree set). As for vertex 15 (1111), \( ! = 9 \) possible combinations are available.

Therefore, we can use a brute force program to find 60 \( \Psi_n \)'s out of \( 2^t \cdot 9^n = 144 \) different \( p(x) \) combinations.

In \( Q_n \), the number of different \( p(x) \) combinations increases to \( 2^{t-1} \cdot 9^n \cdot 4^t = 2660511744 \). We found out 13977216 \( \Psi_n \)'s by running the program.

Since there are such a huge amount of \( \Psi_n \)'s, to enumerate all \( \Psi_n \)'s is impossible for \( n>5 \). Instead, an upper bound for the number of \( \Psi_n \)'s is obtained directly by Lemma 3.1.

**Theorem 1.** An upper bound for the number of \( \Psi_n \)'s is

\[ \prod_{m=3}^{n} \binom{\Psi_m}{\Psi_m} \text{ where } \binom{\Psi_m}{\Psi_m} \text{ denotes the number of } m \text{ 1-bits vertices in } Q_m. \]

**4. A LOWER BOUND OF THE NUMBER OF \( \Psi_N \)**

In this section, we propose an approach for generating (\( n-1 \))! different \( \Psi_n \)'s from a routing square. Then the number of different routing squares forms a lower bound for the number of \( \Psi_n \).

Suppose the binary representation of vertex \( x \) has \( m \) 1-bits. By Lemma 2.3, we can infer that \( x \) is an internal vertex in exactly \( m \) trees of a \( \Psi_n \), and in these trees, \( x \) has a shortest path (with Hamming distance) to the root. Since the number of 1-bits must be reduced vertex by vertex along the path to the root, vertex \( x \) and its ancestors form a decreasing sequence in these trees. That is, if \( p(x) = y \) and \( b(x) = i \), then \( y = x - 2^i \).

A routing square \( [r_{ij}] \) represents the paths from vertex \( 2^n-1 \) to the root in \( n \) spanning tree of a \( \Psi_n \). Every row has \( n \) jumps which stand for the order of changing 1-bit to 0-bits. That is, the \( j \)-th jump in the \( i \)-th tree, denoted by \( r_{ij} \), changes the \( k \)-th bit to 0 where \( k = \log_2 r_{ij} \). Note that the last jump in a row identifies \( T_i \), i.e., \( r_{ij} = 2^{i-1} \).

Based on the binary representation, any positive integer can be decomposed into the sum of distinct powers of 2. Thus, we can also use a jump set to represent a vertex \( x \) (\( 1 \leq x < 2^{n-1} \)) in \( Q_n \). It is obvious that two equivalent jump sets represent the same vertex. Further, two jump sets with different cardinalities must represent different vertices. To meet the requirement of internally disjoint paths, the jump set of the first \( p \) jumps in each of a routing square must be different from each other for \( 1 \leq p < n \). Precisely, \( \{ r_{i,0} : 1 \leq h \leq p \} \neq \{ r_{j,0} : 1 \leq h \leq p \} \) for \( 1 \leq i \neq j \leq n \). We call it the prefix subset unequal property for a routing square.

The first row of a basic routing square is arranged in \( 2^1, 2^2, ..., 2^{n-1}, 2^0 \). For example, there are 10 basic routing squares for \( Q_4 \). In addition to the one of Figure 1, others are listed as follows:

\[
\begin{array}{cccc}
2 & 4 & 8 & 1 \\
1 & 8 & 4 & 2 \\
8 & 2 & 1 & 4 \\
4 & 1 & 2 & 8 \\
\end{array}
\]

\[
\begin{array}{cccc}
2 & 4 & 8 & 1 \\
8 & 2 & 1 & 4 \\
4 & 1 & 2 & 8 \\
\end{array}
\]

Any basic routing square can stand for (\( n-1 \))! routing squares by consecutively swapping two elements of set \( \{ 2^1, 2^2, ..., 2^{n-1} \} \). For example, consecutive swaps (2,4), (2,8), (4,8), (2,4) and (2,8) transform the basic routing square of Figure 1 to the following 5 routing squares, respectively:

\[
\begin{array}{cccc}
4 & 2 & 8 & 1 \\
8 & 1 & 4 & 2 \\
2 & 8 & 1 & 4 \\
4 & 1 & 2 & 8 \\
\end{array}
\]

Obviously, all routing squares with a basic routing square are distinct and satisfy the prefix subset unequal property. Further, any two routing square sets with respect to different basic routing squares are mutually exclusive.

A routing square is able to represent a \( \Psi_n \) because all parent-pending vertices can determine their parents in each spanning tree according to the routing square and satisfy the internally disjoint requirement.

**Lemma 4.1.** For a parent-pending vertex \( x \), a routing square can determine a \( \Psi_n \).

**Proof.** Let \( x \) be a parent-pending vertex for a \( \Psi_n \) and let \( m \) be the number of 1-bits in \( x \). By Corollary 3.2, we have \( 3 \leq m \leq n \). We prove this lemma by principle of mathematical induction on \( m \).

(Base step) For \( m = n \) (\( x = 2^n-1 \)), a routing square \( [r_{ij}] \) can be used to generate \( n \) internally disjoint paths from \( x \) to the root in \( T_i \) (\( i = 1, 2, ..., n \)) due to its prefix subset unequal property.

(Induction step) Suppose \( m = k \) and there exist \( n \) internally disjoint paths from \( x \) to the root in \( T_i \) (\( i = 1, 2, ..., n \)). We can obtain \( k \) paths from the internal vertex \( x \) to the root according to a \( k \) by \( k \) routing square, namely \( [r^*_{ij}] \). For a vertex \( y \) with \( m = k-1 \), there must exists a row in some \( [r^*_{ij}] \) (exactly \( n-k \) of \([r^*_{ij}]\)'s), namely the \( t \)-th row, such that \( y = x - r^*_{it} \), and jumps \( r^*_{i2}, r^*_{i3}, ..., r^*_{ik} (=2^{i-1}) \) can be used to generate a path from \( y \) to the root in \( T_i \).

Then we can assign \( p_i(y) = y - r^*_{ij} \), where \( q = \ldots \)
The $k-1$ paths from $y$ to the root are internally disjoint since they satisfy the prefix subset unequal property. As for the remained $n-k+1$ paths from the leaf vertex $y$ are also internally disjoint since the prefix subset unequal property is held. In case one path from the internal vertex $y$ and another from the leaf vertex $y$, two paths are proved to be internally disjoint because they also satisfy the prefix subset unequal property. \[\text{Q.E.D.}\]

Since we have to compute routing squares $O(2^n)$ times, the time complexity of transforming a routing square to a $\mathcal{G}_n$ is $O(n^2 2^n)$.

Let $R_n$ denote the number of routing squares for $Q_n$. Then $R_4 = 10! = 60$ that is exactly the number of $S_4$. However, $R_5 = 3236! = 77664$ is less than $13977216$, the number of $S_5$. It turns out that only a very small part of all $\mathcal{G}_n$'s can be found by means of routing squares when $n > 4$. Anyway, $R_n$ is a lower bound of the number of $\mathcal{G}_n$'s.

**Theorem 2.** A lower bound for the number of $\mathcal{G}_n$'s is $R_n$.

**5. CONCLUSION REMARKS**

In this paper, we study the properties of OIST on hypercubes and prove an upper bound of the number of OIST solutions is $\prod_{m=3}^{n} (\text{l}m)^{f(m)}$, where $m$ is the number of 1-bits in a vertex, 'l'm is the derangement number of $m$, and $f(m)$ denotes the number of $m$ 1-bits vertices in $Q_n$.

Then, we give the definition of routing squares for $Q_n$ and prove that a routing square can be transformed to an OIST solution. In [11], the authors prove the proposed algorithm by means of Latin squares. However, a routing square is not necessarily to be a Latin square, and vice versa. Like Latin squares, the number of routing square is not known for $n > 5$. Anyway, it forms a lower bound of the number of OIST solutions. A more efficient algorithm for transforming a routing square to an OIST solution is meaningful.

Since there are so huge amount of OIST solutions, the study of fault-tolerant broadcasting on hypercubes become very promising.

**REFERENCES**


