Node-to-node and Node-to-set Disjoint Paths Problems in Bicubes

Arata KANEKO†, Htoo Htoo Sandi KYAW†, Nonmembers, Kunihiro FUJIYOSHI†, and Keiichi KANEKO†‡, Members

SUMMARY In this paper, we propose two algorithms, B-N2N and B-N2S, that solve the node-to-node and node-to-set disjoint paths problems in the bicube, respectively. We prove their correctness and that the time complexities of the B-N2N and B-N2S algorithms are $O(n^2)$ and $O(n^2 \log n)$, respectively, if they are applied in an $n$-dimensional bicube with $n \geq 5$. Also, we prove that the maximum lengths of the paths generated by B-N2N and B-N2S are both $n+2$. Furthermore, we have shown that the algorithms can be applied in the locally twisted cube, too, with the same performance.

key words: bicube, hypercube, interconnection network, locally twisted cube, massively parallel system, topology

1. Introduction

The hypercube [1] was once a very popular topology for interconnection networks of massively parallel systems, and it has many variants. The bicube [2] is such a topology and it attracts much attention [3]–[7] because it can interconnect the same number of nodes with the same degree as the hypercube while its diameter is almost half of that of the hypercube. In addition, the bicube preserves the property of node symmetry.

In this paper, we propose two algorithms, B-N2N and B-N2S, that solve the node-to-node and node-to-set disjoint paths problems in the bicube, respectively. There is a generic algorithm [8] that solves the problems in cube-based topologies. If we apply it to the problems in an $n$-dimensional bicube ($n \geq 3$), we can generate $n$ node-disjoint paths whose lengths are at most $2n - 1$ in $O(n^3)$ time for both problems. On the other hand, B-N2N generates $n$ node-disjoint paths of lengths at most $n+2$ in $O(n^2)$ time while B-N2S generates $n$ node-disjoint paths of lengths at most $n + 2$ in $O(n^2 \log n)$ time. B-N2N and B-N2S use the algorithm proposed by Bossard and Kaneko [9], which we call H-N2S, because a bicube consists of two hypercubes with bijective or one-to-one edges between them. H-N2S solves the node-to-set disjoint paths problem in the hypercube. Our algorithms, B-N2N and B-N2S, can be applied in the locally twisted cube with the same performance because a locally twisted cube also consists of two hypercubes with bijective edges between them [10].

Given a source node $s$ and a destination node $d$ in a $k$-connected graph, the node-to-node disjoint paths problem is to generate $k$ paths $U_i: s \sim d_i$ $(1 \leq i \leq k)$ such that $U_i$ $(1 \leq i \leq k)$ are node-disjoint except for $s$ and $d$. In addition, given a source node $s$ and a set of $k$ destination nodes $\{d_1, d_2, \ldots, d_k\}$ in a $k$-connected graph, the node-to-set disjoint paths problem is to generate $k$ paths $U_i: s \sim d_i$ $(1 \leq i \leq k)$ such that $U_i$ $(1 \leq i \leq k)$ are node-disjoint except for $s$. The node-to-node disjoint paths problem [11]–[16] and the node-to-set disjoint paths problem [8], [9], [17]–[23] are important issues in parallel and distributed computing as well as the set-to-set disjoint paths problem [18], [24]–[28].

Also, we prove that the maximum lengths of the paths generated by B-N2N and B-N2S are $O(n^2)$ and $O(n^2 \log n)$, respectively. There is a generic algorithm [8] that solves the problems in cube-based topologies. If we apply it to the problems in an $n$-dimensional bicube ($n \geq 3$), we can generate $n$ node-disjoint paths whose lengths are at most $2n - 1$ in $O(n^3)$ time for both problems. On the other hand, B-N2N generates $n$ node-disjoint paths of lengths at most $n+2$ in $O(n^2)$ time while B-N2S generates $n$ node-disjoint paths of lengths at most $n + 2$ in $O(n^2 \log n)$ time. B-N2N and B-N2S use the algorithm proposed by Bossard and Kaneko [9], which we call H-N2S, because a bicube consists of two hypercubes with bijective or one-to-one edges between them. H-N2S solves the node-to-set disjoint paths problem in the hypercube. Our algorithms, B-N2N and B-N2S, can be applied in the locally twisted cube with the same performance because a locally twisted cube also consists of two hypercubes with bijective edges between them [10].

Given a source node $s$ and a destination node $d$ in a $k$-connected graph, the node-to-node disjoint paths problem is to generate $k$ paths $U_i: s \sim d_i$ $(1 \leq i \leq k)$ such that $U_i$ $(1 \leq i \leq k)$ are node-disjoint except for $s$ and $d$. In addition, given a source node $s$ and a set of $k$ destination nodes $\{d_1, d_2, \ldots, d_k\}$ in a $k$-connected graph, the node-to-set disjoint paths problem is to generate $k$ paths $U_i: s \sim d_i$ $(1 \leq i \leq k)$ such that $U_i$ $(1 \leq i \leq k)$ are node-disjoint except for $s$. The node-to-node disjoint paths problem [11]–[16] and the node-to-set disjoint paths problem [8], [9], [17]–[23] are important issues in parallel and distributed computing as well as the set-to-set disjoint paths problem [18], [24]–[28]: given a set of $k$ source nodes $\{s_1, s_2, \ldots, s_k\}$ and a set of $k$ destination nodes $\{d_1, d_2, \ldots, d_k\}$ in a $k$-connected graph, the set-to-set disjoint paths problem is to generate $k$ paths $U_i: s_i \sim d_i$ $(1 \leq i \leq k)$, $\{j_1, j_2, \ldots, j_k\} = \{1, 2, \ldots, k\}$ such that $U_i$ $(1 \leq i \leq k)$ are node-disjoint. Generating disjoint paths in a massively parallel system has many applications. For example, multiple pairs of nodes can establish the full-bandwidth communication over a network simultaneously by using the circuit switching. The circuit switching provides an optimal data transfer performance because it does not require any switching inside the routers of intermediate nodes. Also, the circuit switching does not allow any interference with other communications, ensuring security and privacy. The studies of the node-disjoint paths problems with respect to some cube-based topologies are summarized in Table 1.

In the rest of this paper, we use ‘disjoint’ instead of ‘node-disjoint’ for simplicity.

2. Preliminaries

In this section, we give the definitions of related topics and the properties of the bicube. Generally, we adopt the notations and terminology from the traditional graph theory. For example, a path in a graph $G(V, E)$ is an alternate sequence of nodes and edges: $u_1, (u_1, u_2), u_2, \ldots, u_{l-1}, (u_{l-1}, u_l), u_l$ for $u_i \in V$ $(1 \leq i \leq l)$, and we use a shorthand $u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_l$ or $u_1 \sim u_l$ if the intermediate nodes are not important. The length of a path is the number of edges included in the path. Let us consider two paths $P$: $u \sim v$ and $Q$: $x \sim y$. Then, if $P$ and $Q$ do not have any common node, they are disjoint. If $P$ and $Q$ do not have any common node except for $u(= x)$, they are disjoint except for $u(= x)$. If $P$ and $Q$ do not have any common node except for $u(= x)$ and $v(= y)$, they are...
Table 1  Time complexities and maximum path lengths of node-disjoint paths routing algorithms for constructing $n$ disjoint paths in $n$-dimensional cube-based topologies.

<table>
<thead>
<tr>
<th>topology</th>
<th>diameter</th>
<th>node-to-node</th>
<th>node-to-set</th>
<th>set-to-set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicube</td>
<td>$[(n+1)/2]^1$</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Locally Twisted Cube</td>
<td>$[(n+3)/2]^1$</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Twisted Cube</td>
<td>$[(n+1)/2]$</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Crossed Cube</td>
<td>$[(n+1)/2]$</td>
<td>$O(n^2)$ [12]</td>
<td>$3n - 5$ [12]</td>
<td>—</td>
</tr>
<tr>
<td>Twisted Crossed Cube</td>
<td>$[(n+1)/2]$</td>
<td>$O(n^2)$ [16]</td>
<td>$4n - 8$ [16]</td>
<td>—</td>
</tr>
<tr>
<td>0-Möbius Cube</td>
<td>$[(n+2)/2]$</td>
<td>$O(n^2)$ [15]</td>
<td>$3n - 5$ [15]</td>
<td>$O(n^2)$ [22]</td>
</tr>
<tr>
<td>1-Möbius Cube</td>
<td>$[(n+1)/2]$</td>
<td>$O(n^2)$ [15]</td>
<td>$3n - 5$ [15]</td>
<td>$O(n^2)$ [22]</td>
</tr>
</tbody>
</table>

†: $n \geq 7$, ‡: $n \geq 5$, §: $n$ is odd.

**Definition 1**: An $n$-dimensional hypercube, $H_n$, is an undirected graph whose node set is $\{0, 1\}^n$. Given two nodes $u$ and $v$ in $H_n,$ $u$ and $v$ are neighboring if and only if $h(u, v) = 1$ where $h(u, v)$ represents the Hamming distance between $u$ and $v.$

The number of nodes and the diameter of $H_n$ are $2^n$ and $n,$ respectively. $H_n$ is symmetric and its degree is $n.$ $H_n$ has a recursive structure such that it consists of two $H_{n-1}$’s. Also, $H_n$ has a shortest-path routing algorithm SPR that generates one of the shortest paths between any pair of nodes whose length is at most $n$ in $O(n)$ time.

**Definition 2**: Given a bit sequence $u = (u_n, u_{n-1}, \ldots, u_1) \in \{0, 1\}^n$, define a function $h(u)$ by $h(u) = u_n \oplus u_{n-1} \oplus u_{n-2} \oplus \cdots \oplus u_1$ where $\oplus$ represents the exclusive-or operation: $0 \oplus 0 = 1 \oplus 1 = 0$ and $1 \oplus 0 = 0 \oplus 1 = 1$. Then, given a pair of bit sequences $u, v \in \{0, 1\}^n$ with an even $n,$ $u$ and $v$ are in $lp$-relation if and only if $u \neq v$ and $h(u) = h(v) = 0$ or $u = \overline{v}$ and $h(u) = h(v) = 1$.

For instance, for two bit sequences $(0, 1, 0, 1, 0, 1)$ and $(1, 0, 0, 1, 1, 0)$, they are in $lp$-relation because $(0, 1, 0, 0, 1, 1) = (1, 0, 0, 1, 1, 0)$ and $h(0, 1, 0, 0, 1, 1) = h(1, 0, 0, 1, 1, 0) = 1$.

**Definition 3**: An $n$-dimensional bicube, $B_n$ ($n \geq 3$), is an undirected graph whose node set is $\{0, 1\}^n$. Given a node $u = (u_n, u_{n-1}, \ldots, u_1)$ in $B_n$, it has $n$ neighboring nodes $u(i)$ ($1 \leq i \leq n$) where $u(i)$ is defined as $u(i) = u_n \oplus u_{n-1} \oplus \cdots \oplus u_i$. The $u(i)$ is given depending on the parity of $i$.

That is, if $n$ is odd, $u(i)$ is given by $u(i) = (u_n, u_{n-1}, \ldots, u_{i+1}, \overline{u}_{i+1}, u_{i+2}, \ldots, u_1)$ and $u(0)$ is given by $u(0) = (1, 0, 1, 0, \ldots, 0)$.

As an example, $B_5$ is shown in Fig. 1. $B_n$ has the following properties [2]. The number of nodes and the diameter of $B_n$ are $2^n$ and $\left\lfloor \frac{n+1}{2} \right\rfloor$ ($n \geq 7$), respectively. The diameters of $B_3$, $B_4$, $B_5$, and $B_6$ are 3, 4, 4, and 5, respectively. $B_n$ is a node-symmetric graph whose degree is $n$. Because the bicube is bipartite, it does not include any cycle with an odd length. Now, let $B_5^b$ and $B_5^a$ be the subgraphs induced by the node sets $\{(u_n, u_{n-1}, \ldots, u_1) | u_n = 0\}$ and $\{(u_n, u_{n-1}, \ldots, u_1) | u_n = 1\}$, respectively. Then, $B_5^b$ and $B_5^a$ are both isomorphic to an $(n-1)$-dimensional hypercube, $H_{n-1}$. In other words, $B_5$ consists of two $H_{n-1}$’s. The left and right subgraphs, $B_5^b$ and $B_5^a$, form two distinct $H_4$’s in Fig. 1, in which $(0, 0, 1, 1, 1) \in B_5^b$ is connected to $(1, 1, 0, 0, 0) \in B_5^a$ while it is connected to $(1, 0, 1, 1, 1)$ in a 5-dimensional hypercube, $H_5$, for example.

**Lemma 1**: For a node $u \in V(B_n^b)$ ($b \in \{0, 1\}$), there are $n$ paths from $u$ such that the other terminal nodes are included in $V(B_n^b)$, the paths are disjoint except for $u$, and their lengths are at most 2.

(Proof) We can generate $n$ paths $L_i$ ($1 \leq i \leq n$) as follows:

$$L_i : \begin{cases} u \rightarrow u^{(i)} \rightarrow u^{(i, n)} & (1 \leq i \leq n - 1), \\ u \rightarrow u^{(n)} & (i = n). \end{cases}$$

Then, the path lengths are at most 2. $u^{(i, n)}$ ($1 \leq i \leq n - 1$) and $u^{(n)}$ are included in $V(B_n^b)$. Because $u^{(i)}$ ($1 \leq i \leq n$)
n − 1) and \( u^{(n)} \) are distinct neighboring nodes of \( u \), and \( B^0_n \) and \( B^1_n \) are connected by one-to-one edges, the paths are disjoint except for \( u \).

3. **B-N2N Algorithm**

In this section, we describe our algorithm B-N2N that solves the node-to-node disjoint paths problem in an \( n \)-dimensional bicube, \( B_n \). Let \( s \) and \( d \) be the source node and the destination node, respectively. For \( n \) with \( 3 \leq n \leq 4 \), \( B_n \) is isomorphic to \( H_n \). Hence, it is trivial to find \( n \) paths between \( s \) and \( d \) that are disjoint except for \( s \) and \( d \) in \( O(n^2) \) time, whose lengths are at most \( n + 1 \) by using the algorithm proposed by Saad and Schultz [11]. Thus, we assume that \( n \geq 5 \). Without loss of generality, we can assume that \( s \in V(B^0_n) \). Then B-N2N is divided into two cases depending on the position of the destination node.

3.1 **B-N2N Case 1 (\( d \in V(B^0_n) \))**

**Step 1** Apply H-N2S in \( B^0_n \) to generate \((n−1)\) paths \( P_i: s \sim d^{(i)} \) (\( 1 \leq i \leq n−1 \)) that are disjoint except for \( s \).

**Step 2** Select \((n−1)\) edges \( d^{(i)} \to d \) (\( 1 \leq i \leq n−1 \)).

**Step 3** Select two edges \( s \to s^{(i)} \) and \( d^{(n)} \to d \).

**Step 4** Apply SPR in \( B^1_n \) to generate the shortest path \( R: s^{(n)} \sim d^{(n)} \) (Fig. 2).

**Step 5** Construct \( n \) paths, \( U_i \) (\( 1 \leq i \leq n \)), that are disjoint except for \( s \) and \( d \) as follows:

\[
U_i : \begin{cases} \ s \sim d^{(i)} \quad (1 \leq i \leq n−1) \\ s \to s^{(i)} \stackrel{R}{\to} d^{(i)} \to d \quad (i = n) \end{cases}
\]

3.2 **B-N2N Case 2 (\( d \in V(B^1_n) \))**

**Step 1** Select the edge \( s \to s^{(n)} \).

**Step 2** Apply SPR in \( B^1_n \) to generate the shortest path \( R: s^{(n)} \sim d \). Let \( d^{(l)} \) be the neighboring node of \( d \) that is included in \( R \).

**Step 3** Apply H-N2S in \( B^0_n \) to generate \((n−1)\) paths, \( P_i: s \sim d^{(i)} \) (\( 1 \leq i \neq l \leq n−1 \)), that are disjoint except for \( s \) as follows:

\[
P_i : \begin{cases} \ s \sim d^{(i)} \quad (1 \leq i \neq l \leq n−1) \\ s \sim d^{(i)} \quad (i = n) \end{cases}
\]

4. **B-N2S Algorithm**

In this section, we describe our algorithm B-N2S that solves the node-to-set disjoint paths problem in an \( n \)-dimensional bicube, \( B_n \). Let \( s \) be the source node and \( \{d_1, d_2, \ldots, d_n\} \) be the set of \( n \) destination nodes. For \( n \) with \( 3 \leq n \leq 4 \), \( B_n \) is isomorphic to \( H_n \). Hence, it is trivial to find \( n \) paths \( s \sim d_i \) (\( 1 \leq i \leq n \)) that are disjoint except for \( s \) in \( O(n^2) \) time, whose lengths are at most \( n + 1 \) by using the algorithm proposed by Bossard and Kaneko [9]. Thus, we assume that \( n \geq 5 \). Without loss of generality, we can assume that \( s \in V(B^0_n) \) and \( \{d_1, d_2, \ldots, d_n\} \cap V(B^0_n) = \{d_1, d_2, \ldots, d_l\} \).

If \( \{d_1, d_2, \ldots, d_n\} \cap V(B^0_n) = \emptyset \), we execute the steps as \( l = 0 \). Then B-N2S is divided into two cases depending on the distribution of the destination nodes.

4.1 **B-N2S Case 1 (\( l = n \))**

**Step 1** Apply H-N2S in \( B^0_n \) to generate \((n−1)\) paths \( P_i: s \sim d_i \) (\( 1 \leq i \neq l \leq n−1 \)) that are disjoint except for \( s \).

**Step 2** If \( d_n \) is included in one of the paths generated in \( S \), say \( P_n \): \( s \sim d_n \), let \( P_n : s \sim d_n \) by discarding the subpath \( d_n \sim d_{x} \) and exchanging the indices of \( d_x \) and \( d_n \).

**Step 3** Select two edges \( s \to s^{(n)} \) and \( d^{(n)} \to d_n \).

**Step 4** Apply SPR in \( B^1_n \) to generate the shortest path \( R: s^{(n)} \sim d^{(n)} \) (Fig. 4).

**Step 5** Construct \( n \) paths, \( U_i \) (\( 1 \leq i \leq n \)), that are disjoint except for \( s \) and \( d \) as follows:

\[
U_i : \begin{cases} \ s \sim d^{(i)} \quad (1 \leq i \neq l \leq n−1) \\ s \sim d^{(i)} \quad (i = n) \end{cases}
\]

**Step 6** Select \((n−1)\) paths \( Q_i: d^{(i,n)} \to d^{(i)} \to d \) (\( 1 \leq i \neq l \leq n−1 \)) and a path \( Q_n: d^{(n)} \to d \) of length 1 (Fig. 3).
The total time complexity of B-N2N in Case 2 is $O(n^2)$, and the maximum path length is $n + 2$. $U_i$ is disjoint from other $U_i$ except for $s$ as follows:

$$U_i : \begin{cases} \{ s \leadsto d_i \} & (1 \leq i \leq l) \\ s \rightarrow s^{(n)} R \rightarrow d_i & (l + 1 \leq i \leq n - 1) \\ s \rightarrow s^{(n)} R \rightarrow d_i & (i = n) \end{cases}$$

### 5. Correctness and Complexities

**Lemma 2:** For a source node $s$ and a set of $n$ destination nodes $\{d_1, d_2, \ldots, d_n\}$ in an $n$-dimensional hypercube, the H-N2S algorithm by Bossard and Kaneko [9] generates $n$ paths from $s$ to $d_i$ ($1 \leq i \leq n$) of lengths at most $n + 1$ that are disjoint except for $s$ in $O(n^2)$ time. (Proof) From [9].

**Lemma 3:** In Case 1, for a source node $s$ and a destination node $d$ in an $n$-dimensional hypercube with $n \geq 5$, B-N2N takes $O(n^2)$ time to generate $n$ paths from $s$ to $d$ of lengths at most $n + 1$ that are disjoint except for $s$ and $d$. (Proof) In Step 1, H-N2S takes $O(n^2)$ time to generate $(n-1)$ paths $P_i$: $s \leadsto d^{(i)}$ ($1 \leq i \leq n-1$) of lengths at most $n$ that are disjoint except for $s$ from Lemma 2. In Step 2, it takes $O(n)$ time to select $(n-1)$ edges. In Step 3, it takes $O(1)$ time to select two edges. In Step 4, SPR takes $O(n)$ time to generate the shortest path whose length is at most $n + 1$. The total time complexity of B-N2N in Case 1 is $O(n^2)$, and the maximum path length is $n + 1$. The paths $U_i$: $s \leadsto d^{(i)} \rightarrow d$ ($1 \leq i \leq n - 1$) are disjoint except for $s$ and $d$ because $P_i$ ($1 \leq i \leq n - 1$) are generated by H-N2S. The path $U_n$ is disjoint from other paths $U_i$ ($1 \leq i \leq n - 1$) except for $s$ and $d$ because it is outside of $B_n^0$ other than $s$ and $d$. (Proof) In Step 1, it takes $O(1)$ time to select the edge. In Step 2, SPR takes $O(n)$ time to generate the shortest path $P$ of length at most $n - 1$. It takes $O(1)$ time to find $l$ of $d^{(i)}$. In Step 3, H-N2S takes $O(n^2)$ time to generate $(n-1)$ paths $P_i$ ($1 \leq i \neq l \leq n$) of lengths at most $n$ that are disjoint except for $s$ from Lemma 2. In Step 4, it takes $O(n)$ time to generate $Q_i$ ($1 \leq i \neq l \leq n$) of lengths at most 2. The total time complexity of B-N2N in Case 2 is $O(n^2)$, and the maximum path length is $n + 2$. $U_i$ is disjoint from other $U_i$. (Proof) From [9].
(1 ≤ i(̸= l) ≤ n) except for s and d because it is outside of B₀, other than s and d and it does not include any neighboring node of d except for dMarginal. U_i (1 ≤ i(̸= l) ≤ n) are disjoint except for s and d among others because P_i (1 ≤ i(̸= l) ≤ n) are generated by H-N2S and P_i (1 ≤ i(̸= l) ≤ n) include distinct nodes dMarginal, d_i(l) (1 ≤ i(̸= l) ≤ n−1) and dMarginal, d_i(l) (1 ≤ i(̸= l) ≤ n). Consequently, U_i (1 ≤ i ≤ n) are disjoint except for s and d.

**Theorem 1:** For a source node s and a destination node d in an n-dimensional bicube with n ≥ 5, the B-N2N algorithm takes O(n²) time and it generates n paths from s to d of lengths at most n + 2 that are disjoint except for s and d.

(Proof) From Lemmas 3 and 4.

**Lemma 5:** In Case 1, for a source node s and a set of n destination nodes {d₁, d₂, . . . , dₙ} in an n-dimensional bicube with n ≥ 5, B-N2S takes O(n²) time to generate n paths from s to d_i (1 ≤ i ≤ n) of lengths at most n + 1 that are disjoint except for s.

(Proof) In Step 1, H-N2S takes O(n²) time to generate (n − 1) paths P_i; s ∼ d_i (1 ≤ i ≤ n − 1) of lengths at most n that are disjoint except for s from Lemma 2. In Step 2, it takes O(n°C) time to check if d_i is included in one of the paths generated in Step 1. It takes O(n°C) time to discard the subpath and exchange the indices of d_i and d_i−1. In Step 3, it takes O(1) time to select two edges. In Step 4, SPR takes O(n°C) time to generate the shortest path whose length is at most n − 1. The total time complexity of B-N2S in Case 1 is O(n°C), and the maximum path length is n + 1. The paths U_i (1 ≤ i ≤ n − 1) are disjoint except for s because they are generated by H-N2S. The path Uₙ is disjoint from other paths U_i (1 ≤ i ≤ n − 1) except for s because it is outside of B₀ other than s and dₙ.

**Lemma 6:** In Step 1 of Case 2, B-N2S can find the path P_i for each destination node d_i (l + 1 ≤ i ≤ n).

(Proof) From Lemma 1, there are n candidate paths for P_i. Destination nodes d_i (1 ≤ j ≤ l or i + 1 ≤ j ≤ n) can block at most one of them. Also, each of the paths P_i, dMarginal, d_j (l + 1 ≤ j ≤ i − 1) can block at most one of the candidate paths because there is no cycle of length 3 in B_n. Note that because B₀ and B₁ are connected by bijective edges, the edge dMarginal, d_j → d_i of P_i cannot block two candidate paths at a time. Hence, B-N2S can find at least one of n candidates for P_i.

**Lemma 7:** In Case 2, for a source node s and a set of n destination nodes {d₁, d₂, . . . , dₙ} in an n-dimensional bicube with n ≥ 5, B-N2S takes O(n°C log n) time to generate n paths from s to d_i (1 ≤ i ≤ n) of lengths at most n + 2 that are disjoint except for s.

(Proof) In Step 1, it takes O(n°C log n) time to find P_i (l + 1 ≤ i ≤ n) of lengths at most 2 by using a balanced binary search tree. In Step 2, it takes O(1) time to select the edge. In Step 3, SPR takes O(n°C) time to generate R of length at most n − 1. In Step 4, it takes O(n°C) time to check if R includes a node on the paths generated in Step 1. It takes O(n°C) time to find d_i and discard the subpath, update R, and exchange the indices. In Step 5, it takes O(1) to discard Q_i. In Step 6, H-N2S takes O(n°C) time to generate P_i (1 ≤ i ≤ n − 1) of lengths at most n from Lemma 2. The total time complexity of B-N2S in Case 2 is O(n°C log n), and the maximum path length is n + 2. U_i (1 ≤ i ≤ l) are disjoint among others except for s because they are generated by H-N2S. Also, they are disjoint from U_j (l + 1 ≤ i ≤ n − 1) except for s because P_i (l + 1 ≤ i ≤ n − 1) are generated by H-N2S and Q_i (l + 1 ≤ i ≤ n − 1) are outside of B₀ other than d_i. Moreover, the paths are disjoint from Uₙ except for s because it is outside of B₀ other than s. U_i (l + 1 ≤ i ≤ n − 1) are disjoint among others except for s because P_i (l + 1 ≤ i ≤ n − 1) are generated by H-N2S and Q_i (l + 1 ≤ i ≤ n − 1) are generated in Step 1 such that they are disjoint. In addition, the paths are disjoint from Uₙ except for s because R is outside of B₀ other than s, and it is generated such that it is ensured to be disjoint from Q_i, (l + 1 ≤ i ≤ n − 1) in Step 4. Consequently, U_i (1 ≤ i ≤ n) are disjoint except for s.

**Theorem 2:** For a source node s and a set of n destination nodes {d₁, d₂, . . . , dₙ} in an n-dimensional bicube with n ≥ 5, the B-N2S algorithm takes O(n°C log n) time and it generates n paths from s to d_i (1 ≤ i ≤ n) of lengths at most n + 2 that are disjoint except for s.

(Proof) From Lemmas 5 and 7.

Because an n-dimensional locally twisted cube consists of two (n − 1)-dimensional hypercubes with bijective edges between them, it is trivial that we can apply B-N2N and B-N2S in the locally twisted cube with the same performance.

**Definition 4:** An n-dimensional locally twisted cube, LT_n (n ≥ 3), is an undirected graph whose node set is {0, 1}^n. Given a node u = (u₁, u₂, . . . , uₙ) in LT_n, it has n neighboring nodes u⁽⁽i⁾⁾ (1 ≤ i ≤ n) where u⁽⁽i⁾⁾ = (u₁, u₂, . . . , uᵢ⁻¹, uᵢ+¹, uᵢ−¹, . . . , uₙ) (1 ≤ i ≤ n − 2), u⁽⁽n−1⁾⁾ = (u₁, u₂, . . . , uₙ₋₁, uₙ−₂, . . . , u₁), and u⁽⁽n⁾⁾ = (u₁, u₂, . . . , uₙ−₁).

**Theorem 3:** For a source node s and a destination node d in an n-dimensional locally twisted cube with n ≥ 5, the B-N2N algorithm takes O(n°C) time and it generates n paths from s to d of lengths at most n + 2 that are disjoint except for s and d.

(Proof) From Theorem 1.

**Theorem 4:** For a source node s and a set of n destination nodes {d₁, d₂, . . . , dₙ} in an n-dimensional locally twisted cube with n ≥ 5, the B-N2S algorithm takes O(n°C log n) time and it generates n paths from s to d_i (1 ≤ i ≤ n) of lengths at most n + 2 that are disjoint except for s.

(Proof) From Theorem 2.

6. Conclusion

In this paper, we have proposed two algorithms, B-N2N and B-N2S, that solve the node-to-node and node-to-set disjoint
paths problems in the bicube, respectively. We have proved the time complexities of the B-N2N and B-N2S algorithms are $O(n^2)$ and $O(n^2 \log n)$, respectively, and the maximum path lengths are both $n + 2$ if they are applied in an $n$-dimensional bicube with $n \geq 5$. In addition, we have shown that the algorithms can be applied in the locally twisted cube with the same performance.

One of our future works is to check whether the bound of the path lengths $n + 2$ is tight or not. Also, our future works include inventing an algorithm to solve the set-to-set disjoint paths problem in the bicube.

Acknowledgment

We would like to express our sincere gratitude to reviewer B for pointing out insufficient descriptions and suggesting improvements. This study was partly supported by JSPS KAKENHI Grant Number JP23K11029.

References


Arata Kaneko is a master program student at Tokyo University of Agriculture and Technology in Japan. His main research areas are interconnection networks and fault-tolerant systems based on graph theory and network theory. He received the B.E. degree from Tokyo University of Agriculture and Technology in 2022.
Htoo Htoo Sandi Kyaw is an Assistant Professor at Tokyo University of Agriculture and Technology in Japan. Her main research areas are educational technology, web application systems, and graph theory. She received the B.E. and M.E. degrees from University of Technology (Yatanarpon Cyber City) in Myanmar in 2015 and 2018, respectively, and the Ph.D. degree from Okayama University in Japan in 2021.

Kunihiro Fujiyoshi is an Associate Professor at Tokyo University of Agriculture and Technology in Japan. His main research interests are in combinatorial algorithms and VLSI layout design. He received the B.E., M.E., and D.E. degrees from Tokyo Institute of Technology in 1987, 1989, and 1994, respectively. He is a member of IEEE and IPSJ.

Keiichi Kaneko is a Professor at Tokyo University of Agriculture and Technology in Japan. His main research areas are functional programming, parallel and distributed computation, partial evaluation and fault-tolerant systems. He received the B.E., M.E., and Ph.D. degrees from the University of Tokyo in 1985, 1987, and 1994, respectively. He is a member of ACM, IEEE-CS, IPSJ and JSSST.