IEICE TRANS. INF. & SYST., VOL.E99–D, NO.3 MARCH 2016

PAPER

Node-to-Set Disjoint Paths Problem in a Möbius Cube*

David KOCIK†, **Yuki HIRAI**††, ***Nonmembers***, ***and*** **Keiichi KANEKO**††a), ***Member***

SUMMARY This paper proposes an algorithm that solves the node-to-set disjoint paths problem in an n-Möbius cube in polynomial-order time of n. It also gives a proof of correctness of the algorithm as well as estimating the time complexity, $O(n^4)$, and the maximum path length, $2n-1$. A computer experiment is conducted for $n = 1, 2, \ldots, 31$ to measure the average performance of the algorithm. The results show that the average time complexity is gradually approaching to $O(n^3)$ and that the maximum path lengths cannot be attained easily over the range of n in the experiment.

key words: *hypercube, multicomputer, interconnection network, parallel processing, dependable computing*

1. Introduction

Recently, because the clock frequency has shown signs of leveling off, parallel processing systems, especially massively parallel systems are gathering much attention. Though the hypercube [21] was enthusiastically studied in 1980's as a topology for parallel processing systems, it has been seen over the last two decades as obsolete. Recent massively parallel systems adopt the hierarchical topology instead of the conventional simple topologies such as a mesh, a torus, and so on to connect several tens of thousands of processors efficiently. Some massively parallel systems such as NASA Pleiades [19] and ACC Cyfronet AGH Zeus [2], [20] have adopted a hypercube as a higher-layer topology of their hierarchical topologies. Therefore, a hypercube and its variants have returned to the center of attention. A Möbius cube [6] is one such variant of a hypercube.

A Möbius cube can connect the same number of nodes as a hypercube while keeping its diameter about half of that of the hypercube. Hence, it has attracted much attention [8], [14], [23]–[25]. The unsolved problems in Möbius cubes include the node-to-set disjoint paths problem: given a source node $\boldsymbol{s}$ and a set of destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k\}$ in a k-connected graph $G = (V, E)$, find k paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \le i \le k)$ between $\boldsymbol{s}$ and each element of D that are node-disjoint except for $\boldsymbol{s}$. Note that in this paper the notations $\boldsymbol{u} \rightsquigarrow \boldsymbol{v}$ and $\boldsymbol{u} \to \boldsymbol{v}$ for two nodes $\boldsymbol{u}$ and $\boldsymbol{v}$ represent a path from $\boldsymbol{u}$ to $\boldsymbol{v}$ and an edge from $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively. The node-to-set disjoint paths problem is an important issue in parallel and distributed computation [5], [10], [15], [16] as well as the node-to-node disjoint paths problem [7], [12], [18], [22] and the set-to-set disjoint paths problem [3], [4], [9], [11].

In general, we can solve the node-to-set disjoint paths problem in polynomial-order time of $|V|$ by using the maximum flow algorithm. However, the complexity of the algorithm is too large for an n-dimensional Möbius cube M_n because it has 2^n nodes. For an n-dimensional hypercube, there is an algorithm that solves the node-to-set disjoint paths problem in $O(n^2)$ time [5]. The maximum length of the paths generated by the algorithm is $n+1$. However, this algorithm is not applicable to Möbius cubes because they do not have some properties that hold in hypercubes. Therefore, it is necessary to invent an applicable algorithm by investigating properties of an M_n. In this paper, we propose an algorithm N2S (node-to-set) with polynomial-order time of n instead of 2^n. Algorithm N2S is comprised of two cases depending on the distribution of the source node and the destination nodes. The algorithm constructs n disjoint paths from the source node to n destination nodes where n is equal to the connectivity of M_n's. We also present the results of an average performance evaluation by a computer experiment.

The rest of this paper is organized as follows. A definition of a Möbius cube as well as other requisite definitions are introduced in Sect. 2. Section 3 explains our algorithm N2S in detail. Section 4 describes a proof of correctness and the theoretical complexities of N2S. Average performance of N2S is reported in Sect. 5. We conclude and give future works in Sect. 6.

2. Preliminaries

A definition of a Möbius cube and three lemmas are introduced in this section.

Definition 1: An n-dimensional Möbius cube M_n has 2^n nodes. A unique n-bit address is assigned to each node. Two nodes $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$ and $\boldsymbol{v}$ are connected if and only if one of the following conditions is satisfied:

$$\boldsymbol{v} = \begin{cases} (u_1, u_2, \ldots, u_{i-1}, \bar{u}_i, u_{i+1}, \ldots, u_n) & (u_{i-1} = 0), \\ (u_1, u_2, \ldots, u_{i-1}, \bar{u}_i, \bar{u}_{i+1}, \ldots, \bar{u}_n) & (u_{i-1} = 1). \end{cases}$$

where $\bar{u}_i$ represents a bit obtained by reverting u_i. Note that

Manuscript received August 21, 2015.
Manuscript revised November 10, 2015.
Manuscript publicized December 14, 2015.

†The author is with Faculty of Information Technology, Czech Technical University in Prague, Thákurova, Prague, Czech Republic.
††The authors are with Institute of Engineering, Tokyo University of Agriculture and Technology, Koganei-shi, 184–8588 Japan.

*This paper is an extended version based on a conference paper [17].

a) E-mail: k1kaneko@cc.tuat.ac.jp (Corresponding author)

DOI: 10.1587/transinf.2015EDP7331

Table 1 Comparison of a 0-M_n and a 1-M_n with other topologies.

	#nodes	degree	diameter	average distance
0-M_n	2^n	n	$\lceil (n+2)/2 \rceil$	†
1-M_n	2^n	n	$\lceil (n+1)/2 \rceil$	†
H_n	2^n	n	n	$n/2$
T_n	2^n	n	$\lceil (n+1)/2 \rceil$	$\to 3n/8$ $(n \to \infty)$ [1]

†: $\le n/3 + [1 - (-1/2)^n]/9 + 1$ [6]

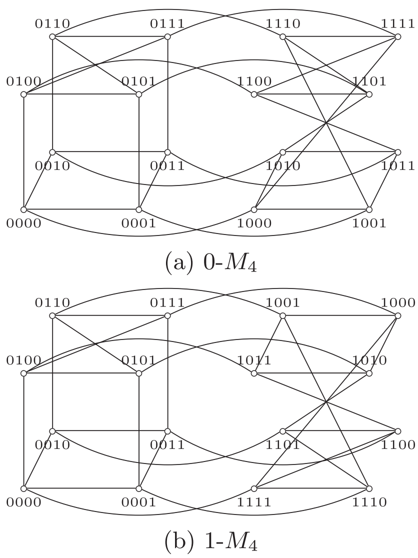

Fig. 1 Examples of a 0-M_4 and a 1-M_4.

u_0 is undefined. Hence, we can assume that $u_0 = 0$ or $u_0 = 1$. The topologies induced by assuming $u_0 = 0$ or $u_0 = 1$ are called a 0-M_n or a 1-M_n, respectively.

If two nodes $\boldsymbol{u}$ and $\boldsymbol{v}$ are connected by one of the conditions in Definition 1, $\boldsymbol{v}$ is denoted by $\boldsymbol{u}^{(i)}$ or $\boldsymbol{u}$ is denoted by $\boldsymbol{v}^{(i)}$. Moreover, if $\boldsymbol{u}_1 = \boldsymbol{u}_0{}^{(i_1)}$, $\boldsymbol{u}_2 = \boldsymbol{u}_1{}^{(i_2)}, \ldots, \boldsymbol{u}_n = \boldsymbol{u}_{n-1}{}^{(i_n)}$ hold, $\boldsymbol{u}_n$ is denoted by $\boldsymbol{u}_0{}^{(i_1,i_2,\ldots,i_n)}$.

Figure 1 shows examples of a 0-M_4 and a 1-M_4. Note that a 0-M_n and a 1-M_n provide different topologies. For example, the average distance for a 0-M_4 is equal to 1.81 while that for a 1-M_4 is equal to 1.75. An M_n is comprised of two disjoint subgraphs M_n^0 and M_n^1 where M_n^i $(i \in \{0, 1\})$ is induced by the set of nodes $\{\boldsymbol{u} = (u_1, u_2, \ldots, u_n) \mid u_1 = i\}$. Note also that an M_n^0 and an M_n^1 are isomorphic to a 0-M_{n-1} and a 1-M_{n-1}, respectively. In addition, neighborhood is not preserved between M_n^0 and M_n^1 in an M_n while it is preserved between two subcubes in a hypercube. That is, for example, two nodes 0110 and 0010 are adjacent in the M_4^0 in a 0-M_4. The nodes are adjacent to the node 1110 and 1010 in the M_4^1, but they are not adjacent. The lack of this property is the major reason why the algorithm proposed by Bossard and Kaneko for hypercubes [5] is not applicable to Möbius cubes.

Table 1 shows a comparison of properties of an n-dimensional 0-Möbius cube, 0-M_n, and an n-dimensional 1-Möbius cube, 1-M_n, with an n-dimensional hypercube, H_n, and an n-dimensional twisted hypercube, T_n, [13]. With respect to the diameter, a T_n has slightly better performance than a 0-M_n. However, a T_n is inferior to a 0-M_n and a 1-M_n regarding the average distance.

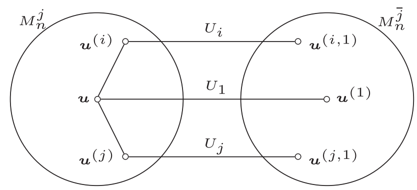

Fig. 2 Disjoint paths between M_n^j and $M_n^{\bar{j}}$.

There is a shortest-path routing algorithm for an arbitrary pair of nodes in an M_n and it takes $O(n)$ time [6]. In the rest of this paper, we refer the algorithm `spr`.

Lemma 1: In an M_n, for an arbitrary node $\boldsymbol{u} \in M_n^j$ $(j \in \{0, 1\})$, there is exactly one edge $\boldsymbol{u} \to \boldsymbol{v} (\in M_n^{\bar{j}})$.

(Proof) Assume $\boldsymbol{u} \in M_n^j$ $(j \in \{0, 1\})$. Then, $\boldsymbol{u}^{(i)} \in M_n^j$ $(2 \le i \le n)$. While $\boldsymbol{u}^{(1)} \in M_n^{\bar{j}}$. Hence, there is exactly one edge $\boldsymbol{u} \to \boldsymbol{u}^{(1)} (\in M_n^{\bar{j}})$. □

Lemma 2: In an M_n, for an arbitrary node $\boldsymbol{u} \in M_n^j$ $(j \in \{0, 1\})$, there are n paths of length at most 2 from $\boldsymbol{u}$ to nodes in the $M_n^{\bar{j}}$ that are disjoint except for $\boldsymbol{u}$.

(Proof) Let us consider n paths from $\boldsymbol{u}$ to nodes in the $M_n^{\bar{j}}$:

$$U_i : \begin{cases} \boldsymbol{u} \to \boldsymbol{u}^{(i)} & (i = 1), \\ \boldsymbol{u} \to \boldsymbol{u}^{(i)} \to \boldsymbol{u}^{(i,1)} & (2 \le i \le n). \end{cases}$$

Then, from Lemma 1, $\boldsymbol{u}^{(1)} \ne \boldsymbol{u}^{(i,1)}$ $(2 \le i \le n)$. Hence, U_1 is disjoint with other paths U_i $(2 \le i \le n)$ except for $\boldsymbol{u}$. In addition, for two paths U_i and U_j $(2 \le i < j \le n)$, from $\boldsymbol{u}^{(i)} \ne \boldsymbol{u}^{(j)}$ and Lemma 1, these paths are also disjoint except for $\boldsymbol{u}$ (Fig. 2). From above discussion, the n paths U_i $(1 \le i \le n)$ of length at most 2 are disjoint except for $\boldsymbol{u}$. □

Lemma 3: There is no cycle whose length is 3 in an M_n.

(Proof) We prove this lemma by induction on n. Clearly, an M_2 does not have a such cycle. Then, we assume that $n \ge 3$ and the lemma holds for an arbitrary M_{n-1}. If an M_n has a cycle of C of length 3, C has at least two edges between the M_n^0 and the M_n^1. Then, from Lemma 1, the terminal nodes of these two edges are all different. Therefore, this fact contradicts that the length of C is 3. □

3. Algorithm N2S

In this section, for a source node $\boldsymbol{s}$ and a set of destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$ in an M_n where $\boldsymbol{s} \notin D$, we show an algorithm N2S that finds n paths R_i: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \le i \le n)$ that are disjoint except for $\boldsymbol{s}$.

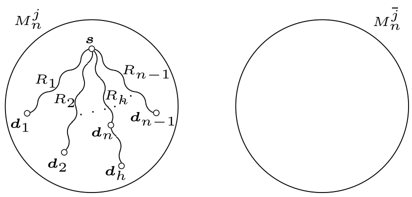

Fig. 3 Case 1, $\boldsymbol{d}_n$ is included in R_h in Step 2 in Algorithm N2S.

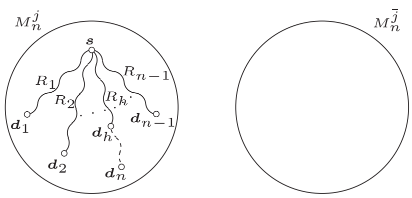

Fig. 4 Case 1, after Step 2 in Algorithm N2S.

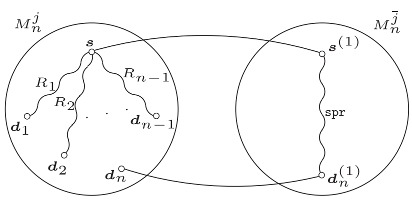

Fig. 5 Case 1, after Step 4 in Algorithm N2S.

Since a 0-M_1 and a 1-M_1 are isomorphic and they consist of two nodes and an edge between them, the solution in a 0-M_1 or a 1-M_1 is trivially $\boldsymbol{s} \to \boldsymbol{d}_1$. Hence, in the rest of the paper, we assume that $n \geq 2$.

3.1 Procedure 1

In case that all destination nodes are included in an M_n^j ($\boldsymbol{s} \in M_n^j$, $D \subset M_n^j$), we construct n paths from $\boldsymbol{s}$ to each destination node in D that are disjoint except for $\boldsymbol{s}$ by the following Procedure 1.

Step 1 In the M_n^j, apply Algorithm N2S recursively to construct $(n-1)$ paths R_i: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq n-1)$ that are disjoint except for $\boldsymbol{s}$.

Step 2 If the node $\boldsymbol{d}_n$ is included in one of the $(n-1)$ disjoint paths constructed in Step 1, say R_h: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_h$ (Fig. 3), discard the subpath $\boldsymbol{d}_n \rightsquigarrow \boldsymbol{d}_h$, and exchange the indices of $\boldsymbol{d}_h$ and $\boldsymbol{d}_n$. See Fig. 4.

Step 3 Select edges $\boldsymbol{s} \to \boldsymbol{s}^{(1)}$ and $\boldsymbol{d}_n \to \boldsymbol{d}_n^{(1)}$.

Step 4 In the $M_n^{\bar{j}}$, construct a path $\boldsymbol{s}^{(1)} \rightsquigarrow \boldsymbol{d}_n^{(1)}$ by using Algorithm `spr`.

Finally, n paths P_i: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq n)$ that are disjoint except for $\boldsymbol{s}$ are constructed as follows:

$$P_i : \begin{cases} \boldsymbol{s} \overset{R_i}{\rightsquigarrow} \boldsymbol{d}_i & (1 \leq i \leq n-1), \\ \boldsymbol{s} \to \boldsymbol{s}^{(1)} \overset{\mathtt{spr}}{\rightsquigarrow} \boldsymbol{d}_n^{(1)} \to \boldsymbol{d}_n & (i = n). \end{cases}$$

See Fig. 5.

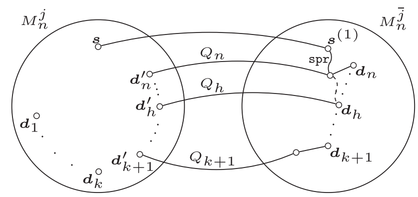

Fig. 6 Case 2, after Step 4 in Algorithm N2S.

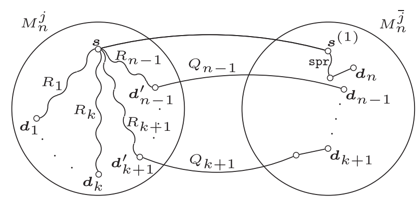

Fig. 7 Case 2, after Step 6 in Algorithm N2S.

3.2 Procedure 2

In case that some destination nodes are included in the $M_n^{\bar{j}}$ ($\boldsymbol{s} \in M_n^j$, $D \cap M_n^{\bar{j}} \neq \emptyset$), we construct n paths from $\boldsymbol{s}$ to each destination node in D that are disjoint except for $\boldsymbol{s}$ by the following Procedure 2.

Step 1 Without loss of generality, we can assume that $D \cap M_n^j = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k\}$ and $D \cap M_n^{\bar{j}} = \{\boldsymbol{d}_{k+1}, \boldsymbol{d}_{k+2}, \ldots, \boldsymbol{d}_n\}$. For $\boldsymbol{d}_{k+1}, \boldsymbol{d}_{k+2}, \ldots, \boldsymbol{d}_n$, from Lemma 2 construct $(n-k)$ mutually disjoint paths Q_i: $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i$ $(k+1 \leq i \leq n)$ of length at most 2 between M_n^j and $M_n^{\bar{j}}$ that do not include $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$. In particular, for each $\boldsymbol{d}_i$ of $\boldsymbol{d}_{k+1}, \boldsymbol{d}_{k+2}, \ldots, \boldsymbol{d}_n$ in this order, select a path Q_i: $\boldsymbol{d}_i \rightsquigarrow \boldsymbol{d}'_i$ among n paths by Lemma 2 that does not include any node on the other paths $Q_{k+1}, Q_{k+2}, \ldots, Q_{i-1}$ or the nodes $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$.

Step 2 Select an edge $\boldsymbol{s} \to \boldsymbol{s}^{(1)}$.

Step 3 In the $M_n^{\bar{j}}$, construct a path $\boldsymbol{s}^{(1)} \rightsquigarrow \boldsymbol{d}_n$ by using Algorithm `spr`.

Step 4 If the path constructed in Step 3 includes any nodes on the paths except for Q_n constructed in Step 1, let $\boldsymbol{d}''_h$ be the node on the path Q_h that is closest to $\boldsymbol{s}^{(1)}$, discard the subpath $\boldsymbol{d}''_h \rightsquigarrow \boldsymbol{d}_n$, and exchange the indices of $\boldsymbol{d}_h$ and $\boldsymbol{d}_n$. See Fig. 6.

Step 5 Discard the subpaths of Q_n except for the subpath constructed in Steps 3 and 4.

Step 6 In the M_n^j, apply Algorithm N2S recursively to construct $(n-1)$ paths R_i: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(2 \leq i \leq k)$ and R_i: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_i$ $(k+1 \leq i \leq n-1)$ that are disjoint except for $\boldsymbol{s}$.

Finally, n paths P_i: $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq n)$ that are disjoint except for $\boldsymbol{s}$ are constructed as follows:

$$P_i : \begin{cases} \boldsymbol{s} \overset{R_i}{\rightsquigarrow} \boldsymbol{d}_i & (1 \leq i \leq k), \\ \boldsymbol{s} \overset{R_i}{\rightsquigarrow} \boldsymbol{d}'_i \overset{Q_i}{\rightsquigarrow} \boldsymbol{d}_i & (k+1 \leq i \leq n-1), \\ \boldsymbol{s} \to \boldsymbol{s}^{(1)} \overset{\mathtt{spr}}{\rightsquigarrow} \boldsymbol{d}_n & (i = n). \end{cases}$$

See Fig. 7.

4. Proof of Correctness and Estimation of Complexities

In this section, we prove the correctness of our algorithm and we give the estimates of the time complexity $T(n)$ and the maximum path length $L(n)$ for an n-dimensional Möbius cube M_n. Proofs are based on induction on n.

We assume that each node can be stored in a machine word, and construction of an edge by obtaining $\boldsymbol{u}^{(i)}$ for any node $\boldsymbol{u}$ requires $O(1)$ time. On the other hand, for any pair of nodes in an M_n, Algorithm `spr` takes $O(n)$ execution time to construct a shortest path between them whose length is at most $\lceil (n+2)/2 \rceil$ [6].

Lemma 4: In an M_n, the paths P_i $(1 \le i \le n)$ constructed by Procedure 1 are disjoint except for $\boldsymbol{s}$. The time complexity of Procedure 1 is $T(n-1) + O(nL(n))$ and the maximum length of the paths constructed is $\max\{L(n-1), \lfloor n/2 \rfloor + 3\}$.
(Proof) The paths P_i $(1 \le i \le n-1)$ constructed in Steps 1 and 2 are disjoint except for $\boldsymbol{s}$ by hypothesis of induction. The path P_n constructed in Steps 3 and 4 is outside of $M_n^{\bar{j}}$ except for $\boldsymbol{s}$ and $\boldsymbol{d}_n$. Hence, P_n cannot share any common node with P_i $(1 \le i \le n-1)$ except for $\boldsymbol{s}$, that is, P_n is disjoint with P_i $(1 \le i \le n-1)$ except for $\boldsymbol{s}$. Step 1 takes $T(n-1)$ time to construct $(n-1)$ paths and the maximum length of them is $L(n-1)$. Step 2 takes $O(nL(n-1))$ time to check whether $\boldsymbol{d}_n$ is included in one of the paths constructed in Step 1. P_n consists of two edges and a subpath by `spr`. Therefore, Steps 2 and 3 take $O(n)$ time to construct a path whose length is at most $2 + \lceil (n+1)/2 \rceil = \lfloor n/2 \rfloor + 3$. Hence, the time complexity of Procedure 1 is $T(n-1)+O(nL(n-1))$ and the maximum path length is $\max\{L(n-1), \lfloor n/2 \rfloor + 3\}$. □

Lemma 5: In an M_n, the paths P_i $(1 \le i \le n)$ constructed by Procedure 2 are disjoint except for $\boldsymbol{s}$. The time complexity of Procedure 2 is $T(n-1) + O(n^3)$ and the maximum length of the paths constructed is $\max\{L(n-1)+2, \lfloor n/2 \rfloor + 2\}$.
(Proof) In Step 1, from Lemma 2, for each $\boldsymbol{d}_i$ of the nodes $\boldsymbol{d}_{k+1}, \boldsymbol{d}_{k+2}, \ldots, \boldsymbol{d}_n$, n paths can be constructed. Each of the nodes $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$ is included at most one of the paths. In addition, from Lemma 3, each of other paths $Q_{k+1}, Q_{k+2}, \ldots, Q_{i-1}$ shares nodes with at most one of the paths given by Lemma 2 for $\boldsymbol{d}_i$. Therefore, at least one of the n paths given by Lemma 2 does not include $\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_k$ or any node on the paths $Q_{k+1}, Q_{k+2}, \ldots, Q_{i-1}$. Hence, in Step 1, $(n-k)$ disjoint paths Q_i $(k+1 \le i \le n)$ of lengths at most 2 can be constructed in $O(n^3)$ time from Lemma 2. The path P_n constructed in Steps 2 to 5 is disjoint with other paths and the length is at most $1 + \lceil (n+1)/2 \rceil = \lfloor n/2 \rfloor + 2$. The time complexity for construction is $O(1) + O(n) + O(n^2) + O(n) = O(n^2)$. The $(n-1)$ paths R_i $(1 \le i \le n-1)$ of lengths at most $L(n-1)$ can be constructed in $T(n-1)$ time in Step 6 and they are disjoint except for $\boldsymbol{s}$ from induction hypothesis. Then, with above discussion, the n paths P_i $(1 \le i \le n)$ are disjoint except for $\boldsymbol{s}$. They can be constructed in $T(n) + O(n^3)$ time and their maximum length is $\max\{L(n-1)+2, \lfloor n/2 \rfloor + 2\}$.□

Theorem 1: For a node $\boldsymbol{s}$ and a set of n nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$ in an M_n, Algorithm N2S finds n paths from $\boldsymbol{s}$ to $\boldsymbol{d}_i$ $(1 \le i \le n)$ that are disjoint except for $\boldsymbol{s}$ in $O(n^4)$ time, and their maximum length is $2n-1$.
(Proof) From $L(1) = 1$ and Lemmas 4 and 5, the constructed paths are disjoint except for $\boldsymbol{s}$ and $L(n) = 2n-1$. Then, $T(n) = O(n^4)$. □

5. Performance Evaluation

We carried out a computer experiment to evaluate average performance of Algorithm N2S. In the experiment, we repeated following steps at least 10,000 times for random pairs of the source $\boldsymbol{s}$ and the set of destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$ in a 0-M_n and a 1-M_n for each n between 1 and 31.

1. Select a set of n destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$ randomly.
2. Select a source node $\boldsymbol{s}$ randomly other than D.
3. For $\boldsymbol{s}$ and D, apply Algorithm N2S and measure the execution time and the maximum path length.

We implemented Algorithm N2S by using the programming language `C++`. The program was compiled with the GNU G++ compiler `g++` with a `-O` option. The target machine is equipped with an Intel Core i5-3230M CPU 2.60 GHz and 4GB RAM. The program was running at Oracle VM VirtualBox with 1GB RAM.

Figures 8 and 9 show the average execution time to construct a node-to-set disjoint paths and their maximum lengths, respectively. Figure 8 shows that the average execution time is gradually approaching to $O(n^3)$ over the range of n in the experiment. From Fig. 9, we can see that the theoretical maximum path length, $2n-1$, is not easily attainable.

Next, we compared difference of performance of Algorithm N2S in a 0-M_n and a 1-M_n. We took the average execution times and the maximum path lengths in a 0-M_n and a 1-M_n for each n shown in Figs. 8 and 9 as samples, and conducted a Wilcoxon rank-sum test. As a result, we could not find any statistically significant difference between them

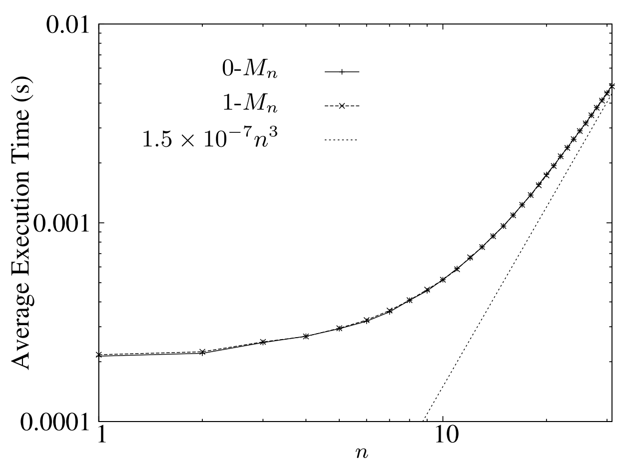

Fig. 8 Average execution time of Algorithm N2S.

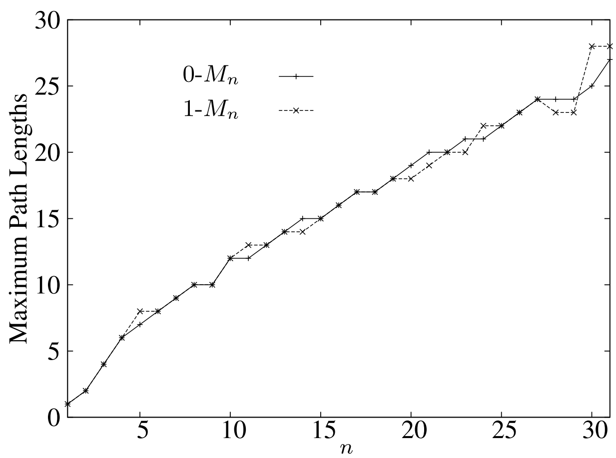

Fig. 9 Maximum lengths of paths constructed by Algorithm N2S.

regarding both of the average execution time ($W = 483$, $p = 0.98$) and the maximum path length ($W = 484.5$, $p = 0.96$). From these results, we can conclude that Algorithm N2S is applicable to a 0-M_n and a 1-M_n equivalently.

6. Conclusions

In this paper, we proposed an algorithm that solves the node-to-set disjoint paths problem in n-Möbius cubes. Theoretical analysis has shown that its time complexity is $O(n^4)$ and the maximum path length is $2n - 1$. We also conducted a computer experiment and showed that the average execution time is gradually approaching to $O(n^3)$ and the maximum path length $2n - 1$ is not easily attainable over the range of n in the experiment.

Future works include theoretical analysis of average performance of the algorithm and improvement of the algorithm to construct shorter paths in smaller execution time.

Acknowledgments

We really appreciate the reviewers for their insightful comments and suggestions. This study is partly supported by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science (JSPS) under Grant No. 25330079.

References

[1] S. Abraham and K. Padmanabhan, "An analysis of the twisted cube topology," Proceedings of the International Conference on Parallel Processing, pp.116–120, Pennsylvania State Press, Aug. 1989.
[2] Academic Computer Center Cyfronet AGH, "Prometheus and Zeus both on TOP500." http://www.cyfronet.krakow.pl/en/15366,artykul,p.html
[3] A. Bossard, "A set-to-set disjoint paths routing algorithm in hyper-star graphs," ISCA International Journal of Computers and Their Applications, vol.21, no.1, pp.76–82, March 2014.
[4] A. Bossard and K. Kaneko, "The set-to-set disjoint-path problem in perfect hierarchical hypercubes," The Computer Journal, vol.55, no.6, pp.769–775, June 2012.
[5] A. Bossard and K. Kaneko, "Time optimal node-to-set disjoint paths routing in hypercubes," Journal of Information Science and Engineering, vol.30, no.4, pp.1087–1093, July 2014.
[6] P. Cull and S.M. Larson, "The möbius cubes," IEEE Transactions on Computers, vol.44, no.5, pp.647–659, May 1995.
[7] M. Dietzfelbinger, S. Madhavapeddy, and I.H. Sudborough, "Three disjoint path paradigms in star networks," Proceedings of the IEEE Symposium on Parallel and Distributed Processing, pp.400–406, Dec. 1991.
[8] J. Fan, "Hamilton-connectivity and cycle-embedding of the möbius cubes," Information Processing Letters, vol.82, no.2, pp.113–117, April 2002.
[9] Q.-P. Gu and S. Peng, "Set-to-set fault tolerant routing in star graphs," IEICE Trans. Inf. & Syst., vol.E79-D, no.4, pp.282–289, April 1996.
[10] Q.-P. Gu and S. Peng, "Node-to-set disjoint paths problem in star graphs," Information Processing Letters, vol.62, no.4, pp.201–207, April 1997.
[11] Q.-P. Gu and S. Peng, "Node-to-set and set-to-set cluster fault tolerant routing in hypercubes," Parallel Computing, vol.24, no.8, pp.1245–1261, 1998.
[12] Y. Hamada, F. Bao, A. Mei, and Y. Igarashi, "Nonadaptive fault-tolerant file transmission in rotator graphs," IEICE Trans. Fundamentals, vol.E79-A, no.4, pp.477–482, April 1996.
[13] P.A.J. Hilbers, M.R.J. Koopman, and J.L.A. van de Snepscheut, "The twisted cube," Volume I: Parallel Architectures on PARLE: Parallel Architectures and Languages Europe, London, UK, vol.258, pp.152–159, Springer-Verlag, 1987.
[14] S.-Y. Hsieh and C.-H. Chen, "Pancyclicity on möbius cubes with maximal edge faults," Parallel Computing, vol.30, no.3, pp.407–421, 2004.
[15] K. Kaneko, "An algorithm for node-to-set disjoint paths problem in burnt pancake graphs," IEICE Trans. Inf. & Syst., vol.E86-D, no.12, pp.2588–2594, Dec. 2003.
[16] K. Kaneko and Y. Suzuki, "An algorithm for node-to-set disjoint paths problem in rotator graphs," IEICE Trans. Inf. & Syst., vol.E84-D, no.9, pp.1155–1163, Sept. 2001.
[17] D. Kocík, Y. Hirai, and K. Kaneko, "An algorithm for node-to-set disjoint paths problem in a möbius cube," Proceedings of the 2015 4th International Student Project Conference, 3B-2, May 2015.
[18] S. Madhavapeddy and I.H. Sudborough, "A topological property of hypercubes: node disjoint paths," Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing, pp.532–539, Dec. 1990.
[19] NASA, "Pleiades supercomputer." http://www.nas.nasa.gov/hecc/resources/pleiades.html.
[20] NOAA, "ESRL GSD media center headlines: New NOAA supercomputer Goes operational." http://esrl.noaa.gov/gsd/media/hotitems/2012/12Apr23.html
[21] C.L. Seitz, "The cosmic cube," Communications of the ACM, vol.28, no.1, pp.22–33, Jan. 1985.
[22] Y. Suzuki and K. Kaneko, "An algorithm for node-disjoint paths in pancake graphs," IEICE Trans. Inf. & Syst., vol.E86-D, no.3, pp.610–615, March 2003.
[23] C.-H. Tsai, "Embedding of meshes in möbius cubes," Theoretical Computer Science, vol.401, no.1-3, pp.181–190, July 2008.
[24] J.-M. Xu, M. Ma, and M. Lü, "Paths in möbius cubes and crossed cubes," Information Processing Letters, vol.97, no.3, pp.94–97, Feb. 2006.
[25] X. Yang, G.M. Megson, and D.J. Evans, "Pancyclicity of möbius cubes with faulty nodes," Microprocessors and Microsystems, vol.30, no.3, pp.165–172, May 2006.

David Kocik is a master program student of Faculty of Information Technology at Czech Technical University in Prague in Czech Republic. His main research areas are graph theory, dependable computing, and fault-tolerant systems. He received the B.E. degree from Czech Technical University in Prague in 2013.

Yuki Hirai is an Assistant Professor at Tokyo University of Agriculture and Technology in Japan. His main research areas are educational technologies and computer-supported collaborative learning. He received the B.L.A. and M.Ed. degrees from Tokyo Gakugei University in 2007 and 2009, respectively. He also received the Ph.D. degree from University of Tsukuba in 2012. He is a member of IPSJ and JSAI.

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.