PAPER

An Algorithm for Node-to-Set Disjoint Paths Problem in Rotator Graphs

Keiichi KANEKO[†], *Regular Member and* Yasuto SUZUKI[†], *Student Member*

SUMMARY In this paper, we give an algorithm for the node-to-set disjoint paths problem in rotator graphs with its evaluation results. The algorithm is based on recursion and it is divided into cases according to the distribution of destination nodes in classes into which all the nodes in a rotator graph are categorized. The sum of the length of paths obtained and the time complexity of the algorithm are estimated and verified by computer simulation.

key words: *rotator graphs, node-to-set disjoint paths problem, interconnection networks, parallel and distributed processing*

1. Introduction

Currently, studies of parallel and distributed computation are becoming more significant. Moreover, research on so-called massively parallel machines has been conducted enthusiastically in recent years. Hence many complex topologies of interconnection networks have been proposed to replace the simple networks such as a hypercube and a mesh [1], [3], [6]. Unfortunately, there still remain unknowns in several metrics for these topologies, making a clear comparison of them difficult. A rotator graph [2] is one of the new topologies that shows promise in that it has a low degree and a small diameter in comparison with the number of nodes. For this topology, algorithms for some problems such as the shortest path routing [2], the Hamiltonian cycle [2], the node-to-node disjoint paths [5], the vertex coloring [8], and the adaptive fault-tolerant routing [9] have already been developed. However, for problems such as the optimal-time broadcasting, the channel graph, and the t-spanners, no good algorithms have been found. The node-to-set disjoint paths problem is one of these unsolved problems: Given a source node $\boldsymbol{s}$ and a set $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_k\}$ ($\boldsymbol{s} \notin D$) of k destination nodes in a k-connected graph $G = (V, E)$, find k paths from $\boldsymbol{s}$ to $\boldsymbol{d}_i$ ($1 \le i \le k$) which are node-disjoint except for $\boldsymbol{s}$. This is one of the most important issues in the design and implementation of parallel and distributed computing systems [4], [7]. In general, node-disjoint paths can be obtained in polynomial order of $|V|$ by making use of the maximum flow algorithm. In an n-rotator graph, the number of nodes is equal to $n!$, so its complexity is not considered efficient. In this paper, we give an answer to this problem which is of polynomial order of n instead of $n!$ and it is evaluated by computer simulation.

The rest of this paper is constructed as follows. Section 2 introduces rotator graphs as well as the notion of classes and explains the problem. Section 3 explains our algorithm in detail and computer simulation is performed in Sect. 4. Section 5 describes conclusions and future works.

Manuscript received December 14, 2000.
Manuscript revised April 11, 2001.

[†]The authors are with the Faculty of Technology, Tokyo University of Agriculture and Technology, Koganei-shi, 184–8588 Japan.

2. Preliminaries

In this section, we first give a definition of a rotator graph, then give some comparisons between a rotator graph and other major network topologies for several elementary indices.

Definition 1: An n-rotator graph, P_n, is a directed graph which has $n!$ nodes. Each node has a unique label $(a_1, a_2, \cdots, a_n)$ comprised of a permutation of n figures: $1, 2, \cdots, n$. In addition, there exists a directed edge $(\boldsymbol{a}, \boldsymbol{b})$ between two nodes $\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$ and $\boldsymbol{b} = (b_1, b_2, \cdots, b_n)$ iff there exists i ($2 \le i \le n$) such that $b_1 = a_2, b_2 = a_3, \cdots, b_{i-1} = a_i, b_i = a_1, b_{i+1} = a_{i+1}, \cdots, b_n = a_n$. Here, let R_i represent the operation to obtain the node $\boldsymbol{b}$ from the node $\boldsymbol{a}$, where $\boldsymbol{a}$ is a parent node of $\boldsymbol{b}$.

An n-rotator graph P_n contains n different $(n-1)$-subrotator graphs. All of the nodes in each subrotator graph P_{n-1} share the same last figure k in their labels and the subrotator graph is specified by $P_{n-1}k$. Then any edge between two nodes which belong to different subrotator graphs $P_{n-1}h$ and $P_{n-1}k$ ($h \ne k$) is given only by the operation R_n. Figure 1 presents some examples of rotator graphs.

Table 1 shows the comparison of an n-rotator graph P_n with an n-star graph S_n, an n-cube Q_n, an (n, k)-de Bruijn graph $B(n, k)$, and an (n, k)-Kautz graph $K(n, k)$. From this table, we can see that the n-rotator graph shows a better performance against the topologies S_n and Q_n in that it can connect more nodes for a given diameter or a given connectivity. Although the n-rotator graph is inferior to the de Bruijn and Kautz graphs, they neither have symmetry nor recursive structure and they are impractical for running applications.

Next, we define a class, which is a subset of nodes,

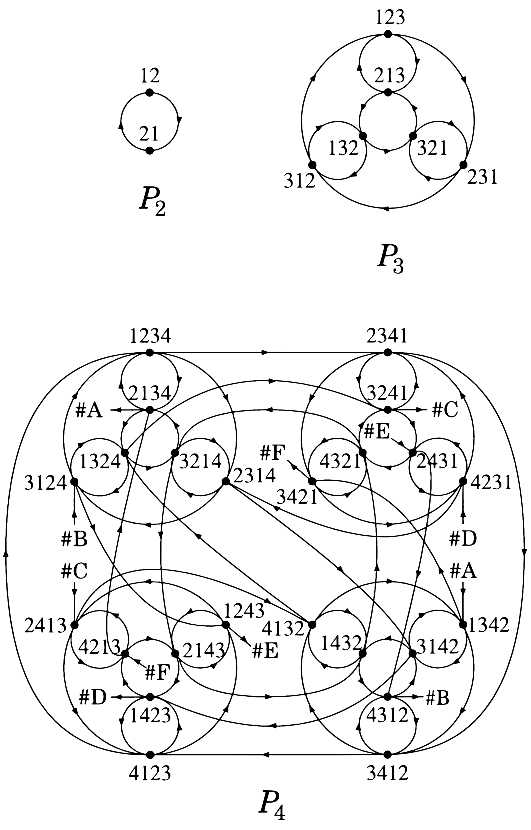


Fig. 1 Examples of rotator graphs.

Table 1 Comparison of a rotator graph with other topologies.

	# nodes	# edges	diameter	connect.
P_n	$n!$	$(n-1)\times n!$	$n-1$	$n-1$
S_n	$n!$	$(n-1)\times n!$	$\left\lfloor \frac{3}{2}(n-1) \right\rfloor$	$n-1$
Q_n	2^n	$n2^n$	n	n
$B(n,k)$	n^k	n^{k+1}	k	n
$K(n,k)$	n^k+n^{k-1}	$n^{k+1}+n^k$	k	n

and the node-to-set disjoint paths problem.

Definition 2: A class of an n-rotator graph is a set of nodes in which for any pair of nodes $\boldsymbol{a}$ and $\boldsymbol{b}$, the node $\boldsymbol{a}$ is obtained from the node $\boldsymbol{b}$ by iterative application of the operation R_n.

The class to which a node $\boldsymbol{a}$ belongs is specified by $C(\boldsymbol{a})$. Lemma 1 shows some properties of classes.

Lemma 1: The following properties hold for classes.

1. Each class has n nodes which form a directed ring structure.
2. Every node in P_n belongs to exactly one class.
3. Every node in P_n has $n-1$ parent nodes, all of which belong to different classes.
4. All the nodes in each subrotator graph belong to different classes.

Proof: Each property is proved as follows:

1. For a node $\boldsymbol{a} = (a_1, a_2, \cdots, a_n)$, a node set $\{(a_2, a_3, \cdots, a_n, a_1), (a_3, a_4, \cdots, a_n, a_1, a_2), \cdots, (a_n, a_1, \cdots, a_{n-1}), (a_1, a_2, \cdots, a_n)\}$ is obtained by repetitive applications of the operation R_n to $\boldsymbol{a}$. The node set forms a cycle of length n. All nodes in this set have different last figures in their lables. Hence, these nodes belong to different subrotator graphs, and they form a simple cycle, or a ring.
2. Assume that a node $\boldsymbol{a}$ belongs to two classes C_1 and C_2. Then from Definition 2 all nodes in the class C_1 belong to C_2 and vice versa. Hence $C_1 = C_2$.
3. All parent nodes of a node $\boldsymbol{a}$, except for one that belongs to the class $C(\boldsymbol{a})$, belong to the same subrotator graph as $\boldsymbol{a}$. Let us call the node in $C(a)$, $\boldsymbol{b}$. From the proof of property 1, all the parent nodes of $\boldsymbol{a}$, except for $\boldsymbol{b}$, and the node $\boldsymbol{a}$ belong to different classes. The node $\boldsymbol{b}$ belongs to the same class as the node $\boldsymbol{a}$. Hence this property holds.
4. From the proof of property 1, all nodes in each class belong to different subrotator graphs. Therefore this property also holds. □

Finally we define the node-to-set disjoint paths problem.

Definition 3: The node-to-set disjoint paths problem in an n-rotator graph is to find $n-1$ paths from a source node $\boldsymbol{s}$ to each node in the destination node set $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_{n-1}\}$ which are disjoint except for $\boldsymbol{s}$.

3. Algorithm

In this section we give an algorithm for the node-to-set disjoint paths problem in an n-rotator graph and it is proved by induction with respect to n. Its basic idea is as follows:

- If all destination nodes belong to different classes, we can construct $n-2$ disjoint paths from the source node to $n-2$ destination nodes by first applying our algorithm recursively in a subrotator graph to obtain $n-2$ disjoint paths from the source node to $n-2$ nodes that belong to the same classes as the $n-2$ destination nodes and then traversing rings of classes to those destination nodes. For the remaining destination node, it is possible to construct a path that is disjoint from other $n-2$ paths by making use of techniques such as the node-to-node disjoint paths algorithm. (See Case II below.)
- If multiple destination nodes belong to one or more classes, this case is reducible to the previous one. We can do this by finding parent nodes for all the destination nodes, except for one, in each class in condition that the classes of the parent nodes are different from classes that include other parent nodes or desination nodes. (See Case I below.)

Because of the symmetry of P_n, we can fix the source node $\boldsymbol{s}$ to be $(1, 2, \cdots, n)$ without any loss of generality. For a 2-rotator graph P_2, the problem is trivial and we assume that $n > 2$ in the following. Let $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_{n-1}\}$ represent the set of destination nodes. Our algorithm is composed of procedures corresponding to the cases given below.

Case I: There exists at least one single class with multiple destination nodes in P_n.
Case II: Each class in P_n has at most one destination node.

- Case II-1: The class $C(\boldsymbol{s})$ includes exactly one destination node.
- Case II-2: $C(\boldsymbol{s})$ has no destination node.
    - Case II-2-A: All the destination nodes in P_n belong to $P_{n-1}n$.
    - Case II-2-B: At least one destination node in P_n belongs to a subrotator graph other than $P_{n-1}n$.
        - Case II-2-B-a: Each subrotator graph of P_n other than $P_{n-1}n$ has at most one destination node.
        - Case II-2-B-b: There exists at least one subrotator graph of P_n other than $P_{n-1}n$ which includes multiple destination nodes.

The following subsections present procedures for the leaf cases, that is, Case I, Case II-1, Case II-2-A, Case II-2-B-a and Case II-2-B-b, as well as proofs of their correctness.

3.1 Case I

In this subsection, we will consider the case that there exist multiple destination nodes in a single class. Let $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$ $(k < n-1)$ be the collection of classes to which destination nodes belong. Using Procedure 1 below, we can construct $n-1$ paths from the source node $\boldsymbol{s}$ to the $n-1$ destination nodes in D which are disjoint except for $\boldsymbol{s}$.

Procedure 1:

1. Let D_1 be a node set constructed by choosing one destination node from each class in $\mathcal{C}$ such that the figure n occurs in the label of the node in the right most position compared to all the other destination nodes in the class. That is, any destination node in D_1 is nearest from the subrotator graph $P_{n-1}n$ with respect to the operation R_n amongst the other destination nodes which belong to the same class as the destination. Additionally, let D_2 be the rest of the destination nodes $D - D_1$. Then, $|D_1| = k, |D_2| = n-k-1$. Without loss of generality, we can assume that $D_2 = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_{n-k-1}\}$. See Fig. 2.
2. For each node $\boldsymbol{d}_i$ in $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_{n-k-2}\}$, select a parent node $\boldsymbol{c}_i$ satisfying the following two conditions in a greedy manner. Note that the number of parent nodes of each $\boldsymbol{d}_i$ that belong to different classes from $C(\boldsymbol{d}_i)$ is equal to $n-2$, which is greater than $n-k-2$.

   Condition 1: If $i \neq j$, $C(\boldsymbol{c}_i) \neq C(\boldsymbol{c}_j)$.
   Condition 2: $\forall i$, $C(\boldsymbol{c}_i) \notin \mathcal{C}$.

   If we also select a parent node of the node $\boldsymbol{d}_{n-k-1}$, the probability increases that case (a), which is of smaller complexity, is selected instead of case (b) in step 4 below. However, the average gain is very small in comparison to the average cost incurred in the selection of the parent. So, the node $\boldsymbol{d}_{n-k-1}$ is not considered in this step.
3. Let $D_1 \leftarrow D_1 \cup \{\boldsymbol{c}_i | 1 \leq i \leq n-k-2\}$, and $\mathcal{C} \leftarrow \mathcal{C} \cup \{C(\boldsymbol{c}_i) | 1 \leq i \leq n-k-2\}$. Then, $|\mathcal{C}| = n-2$.
4. a. In the case that $C(\boldsymbol{s}) \in \mathcal{C}$: (See Fig. 3).
    - i. For a node $\boldsymbol{d}_{n-k-1}$, select its parent node $\boldsymbol{c}_{n-k-1}$ which satisfies the following condition in a greedy manner.

      Condition: $C(\boldsymbol{c}_{n-k-1}) \notin \mathcal{C}$.
    - ii. Let $D_1 \leftarrow D_1 \cup \{\boldsymbol{c}_{n-k-1}\}$, and $\mathcal{C} \leftarrow \mathcal{C} \cup C(\boldsymbol{c}_{n-k-1}) - C(\boldsymbol{s})$. See Fig. 4.
    - iii. Let $\boldsymbol{v}_i$ $(1 \leq i \leq n-2)$ be nodes which belong to the classes in $\mathcal{C}$ and also belong

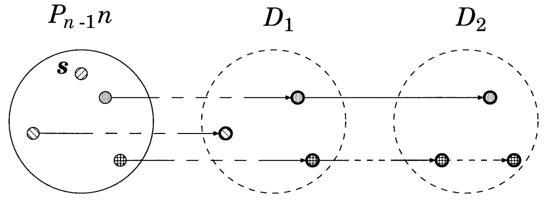


Fig. 2 The sets D_1 and D_2.

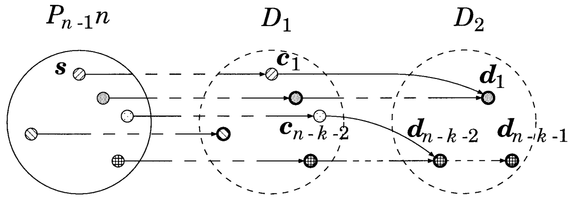

$P_{n\text{-}1}n$
D_1
D_2
$\boldsymbol{s}$
$\boldsymbol{c}_1$
$\boldsymbol{d}_1$
$\boldsymbol{c}_{n\text{-}k\text{-}2}$
$\boldsymbol{d}_{n\text{-}k\text{-}2}$
$\boldsymbol{d}_{n\text{-}k\text{-}1}$

Fig. 3 Case $C(\boldsymbol{s}) \in \mathcal{C}$.

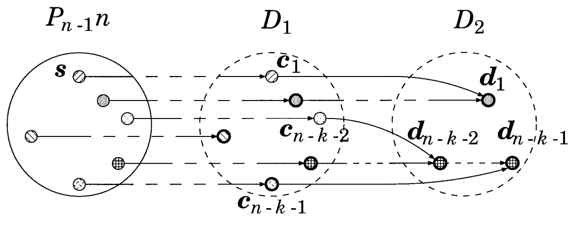

$P_{n\text{-}1}n$
D_1
D_2
$\boldsymbol{s}$
$\boldsymbol{c}_1$
$\boldsymbol{d}_1$
$\boldsymbol{c}_{n\text{-}k\text{-}2}$
$\boldsymbol{d}_{n\text{-}k\text{-}2}$
$\boldsymbol{d}_{n\text{-}k\text{-}1}$
$\boldsymbol{c}_{n\text{-}k\text{-}1}$

Fig. 4 Selection of a parent node c_{n-k-1}.

to the subrotator graph $P_{n-1}n$.

iv. In $P_{n-1}n$, obtain $n-2$ paths from $\boldsymbol{s}$ to $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ which are disjoint except for $\boldsymbol{s}$ by calling the algorithm recursively.

v. Establish paths from $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ and $\boldsymbol{s}$ to corresponding nodes in D_1 within the classes.

vi. Select edges $\boldsymbol{c}_i \to \boldsymbol{d}_i$ $(1 \le i \le n-k-1)$. See Fig. 5.

b. In the case that $C(\boldsymbol{s}) \notin \mathcal{C}$: (See Fig. 6).

i. Let $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ be the nodes which belong to the classes in $\mathcal{C}$ and also belong to the subrotator graph $P_{n-1}n$.

ii. In $P_{n-1}n$, obtain $n-2$ paths from the node $\boldsymbol{s}$ to $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ which are disjoint except for $\boldsymbol{s}$ by calling the algorithm recursively.

iii. Establish paths from $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ to corresponding nodes in D_1 within the classes.

iv. Select edges $\boldsymbol{c}_i \to \boldsymbol{d}_i$ $(1 \le i \le n-k-2)$. See Fig. 7.

v. Let $P_{n-1}l$ be the subrotator graph which includes $\boldsymbol{d}_{n-k-1}$.

vi. Let $\tilde{\boldsymbol{s}}$ represent a node which belongs to $C(\boldsymbol{s})$ and $P_{n-1}l$. Then we can select a path from $\boldsymbol{s}$ to the node $\tilde{\boldsymbol{s}}$ within the class $C(\boldsymbol{s})$.

vii. Let $\boldsymbol{u}_i$ $(1 \le i \le n-2)$ represent the nodes which belong to the classes in $\mathcal{C}$ and also belong to the subrotator graph $P_{n-1}l$. Note that $\boldsymbol{d}_{n-k-1}$ is included in $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_{n-2}\}$.

viii. In $P_{n-1}l$, construct $n-2$ internally disjoint paths from $\tilde{\boldsymbol{s}}$ to $\boldsymbol{d}_{n-k-1}$ [5]. Select a path $\tilde{\boldsymbol{s}} \to \boldsymbol{d}_{n-k-1}$ among them which does not include none of $n-3$ nodes $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_{n-2}\} - \{\boldsymbol{d}_{n-k-1}\}$. See Fig. 8.

Lemma 2: The $n-1$ paths $\boldsymbol{s} \to \boldsymbol{d}_i$ $(1 \le i \le n-1)$ established in Procedure 1 are disjoint except for $\boldsymbol{s}$. Additionally, the sum of the length of paths established in Procedure 1 excluding those which are constructed by the recursive call of this algorithm is of $O(n^2)$.

Proof: The proof is divided into two cases depending on whether $C(\boldsymbol{s})$ is included in $\mathcal{C}$ or not.

1. $C(\boldsymbol{s}) \in \mathcal{C}$
The paths selected in step 4(a)iv of Procedure 1 are disjoint except for $\boldsymbol{s}$ by the induction hypothesis. The $n-1$ paths selected in step 4(a)v are known to be disjoint because of property 2 of classes. The paths selected in step 4(a)v and the edges in step 4(a)vi are apparently disjoint except for the nodes $\boldsymbol{c}_i$ $(1 \le i \le n-k-1)$. All the nodes which are used to construct the paths selected in step 4(a)iv have the figure n in the last position of their labels. In addition, all the nodes which are used to construct the paths selected in step 4(a)v and the edges selected in step 4(a)vi have last figures other than n in their labels except for $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ and $\boldsymbol{s}$. Hence, the paths selected in step 4(a)iv and the paths and edges selected in steps 4(a)v and 4(a)vi are trivially disjoint. Therefore, if $C(\boldsymbol{s})$ is in the collection $\mathcal{C}$, the $n-1$ paths constructed by Procedure 1 are disjoint except for node $\boldsymbol{s}$.
In addition, the sum of the length of $n-1$ paths established in step 4(a)v is of $O(n^2)$. The sum of the length of $n-k-1$ edges selected in step 4(a)vi is of $O(n)$. Summing up them, in the case of $C(\boldsymbol{s}) \in \mathcal{C}$, the sum of the length of paths established in Procedure 1 excluding those which are constructed by the recursive call of the algorithm is of $O(n^2)$.

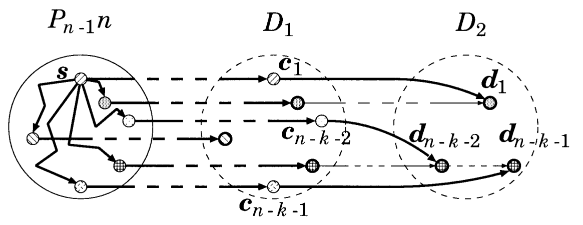


Fig. 5 Establishment of paths.

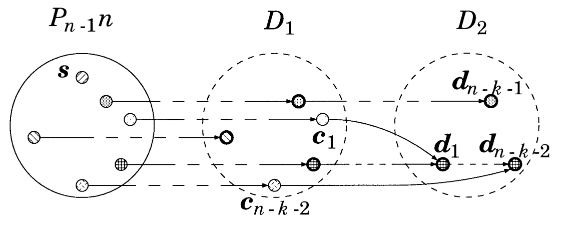


Fig. 6 Case $C(\boldsymbol{s}) \notin \mathcal{C}$.

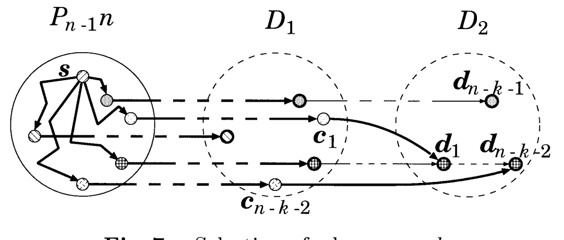


Fig. 7 Selection of edges $c_i \to d_i$.

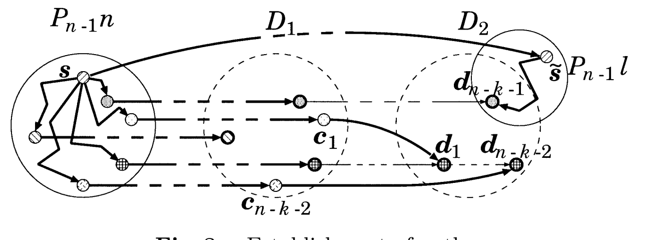


Fig. 8 Establishment of paths.

2. $C(\boldsymbol{s}) \notin \mathcal{C}$
   Following similar reasoning as in the case of $C(\boldsymbol{s}) \in \mathcal{C}$, the paths selected in steps 4(b)ii to 4(b)iv are disjoint except for $\boldsymbol{s}$. The $n-2$ paths selected in steps 4(b)ii to 4(b)iv and the edges selected in step 4(b)vi are known to be disjoint except for $\boldsymbol{s}$ because of property 2 of classes. The path $\tilde{\boldsymbol{s}} \to \boldsymbol{d}_{n-k-1}$ selected in step 4(b)viii and the paths selected in steps 4(b)ii to 4(b)vi are disjoint except for $\tilde{\boldsymbol{s}}$. Therefore, if the class $C(\boldsymbol{s})$ is not in the collection $\mathcal{C}$, the $n-1$ paths selected in Procedure 1 are disjoint except for $\boldsymbol{s}$.
   Moreover, the sum of the length of $n-2$ paths established in step 4(b)iii is of $O(n^2)$. The sum of the length of $n-k-2$ edges selected in step 4(b)iv is of $O(n)$. In addition, the length of each path selected in steps 4(b)vi and 4(b)viii is of $O(n)$. Summing up them, in the case of $C(\boldsymbol{s}) \notin \mathcal{C}$, the sum of the length of paths established in Procedure 1 excluding those which are constructed by the recursive call of the algorithm is of $O(n^2)$. □

3.2 Case II-1

In this subsection, we will consider the case that there is at most one destination node in each class and there is a destination node in the class to which the source node $\boldsymbol{s}$ belongs. By Procedure 2 below, we can construct $n-1$ paths from $\boldsymbol{s}$ to $n-1$ nodes in D which are disjoint except for $\boldsymbol{s}$.

Procedure 2:

1. Let $\tilde{\boldsymbol{b}}$ be the destination node which belongs to the class $C(\boldsymbol{s})$.
2. Select paths within the class $C(\boldsymbol{s})$ from $\boldsymbol{s}$ to $\tilde{\boldsymbol{b}}$.
3. Let $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ represent the nodes in $P_{n-1}n$ and also in the classes to which the $n-2$ nodes in the set $D - \{\tilde{\boldsymbol{b}}\}$ belong.
4. Obtain the paths from $\boldsymbol{s}$ to $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ which are disjoint except for $\boldsymbol{s}$ by calling the algorithm recursively.
5. Select a path within the class from each node $\boldsymbol{v}_i$ $(1 \le i \le n-2)$ to the corresponding node in $D - \{\tilde{\boldsymbol{b}}\}$.

Lemma 3: The $n-1$ paths, $\boldsymbol{s} \to \boldsymbol{d}_i$ $(1 \le i \le n-1)$, selected in Procedure 2 are disjoint except for $\boldsymbol{s}$. Additionally, the sum of the length of paths established in Procedure 2 excluding those which are constructed by the recursive call of this algorithm is of $O(n^2)$.

Proof: The paths selected in step 4 of Procedure 2 are disjoint except for $\boldsymbol{s}$ by the induction hypothesis. The paths selected in step 2 and the paths selected in step 5 are disjoint because of property 2 of classes. All the nodes on the paths selected in step 4 have the figure n in the last position of their labels. In addition, all the nodes which are used to construct paths selected in steps 2 and 5 have last figures other than n in their labels, except for $\boldsymbol{s}$ and $\boldsymbol{v}_i$ $(1 \le i \le n-2)$. Hence, the paths selected in step 4 and the paths selected in steps 2 and 5 are disjoint except for $\boldsymbol{s}$ and $\boldsymbol{v}_i$ $(1 \le i \le n-2)$. Therefore, the $n-1$ paths constructed in Procedure 2 are disjoint except for the source node $\boldsymbol{s}$.

Moreover, the length of path selected in step 2 is of $O(n)$ and the sum of the length of paths selected in step 5 is of $O(n^2)$. Summing up them, the sum of the length of paths established in Procedure 2 excluding those which are constructed by the recursive call of the algorithm is of $O(n^2)$. □

3.3 Case II-2-A

In this subsection, we will consider the case that there is at most one destination node in each class, there is no destination node in the class to which the source node $\boldsymbol{s}$ belongs and all the destination nodes belong to the subrotator graph $P_{n-1}n$ to which the source node $\boldsymbol{s}$ belongs. Procedure 3 below gives the $n-1$ paths from $\boldsymbol{s}$ to $n-1$ destination nodes in D which are disjoint except for $\boldsymbol{s}$.

Procedure 3:

1. Let $D_1 = \{\boldsymbol{d}_i | 1 \le i \le n-2\}$.
2. Obtain $n-2$ paths from $\boldsymbol{s}$ to $n-2$ destination nodes in D_1 which are disjoint except for the node $\boldsymbol{s}$ by calling the algorithm recursively. Here, if the destination node $\boldsymbol{d}_{n-1}$ is on one of the paths obtained, say, a path from $\boldsymbol{s}$ to a node $\boldsymbol{d}_k$, then exchange the specifications of nodes $\boldsymbol{d}_{n-1}$ and $\boldsymbol{d}_k$.
3. Let $\boldsymbol{v}$ represent a node in $P_{n-1}1$ which belongs to the class $C(\boldsymbol{d}_{n-1})$.
4. Select an edge $\boldsymbol{s} \to (2, 3, \cdots, n, 1)$.
5. In $P_{n-1}1$, establish the shortest path from $(2, 3, \cdots, n, 1)$ to $\boldsymbol{v}$.
6. Construct a path from $\boldsymbol{v}$ to the destination node $\boldsymbol{d}_{n-1}$ within the class. See Fig. 9.

Lemma 4: The $n-1$ paths $\boldsymbol{s} \to \boldsymbol{d}_i$ $(1 \le i \le n-1)$ selected in Procedure 3 are disjoint except for $\boldsymbol{s}$. Additionally, the sum of the length of paths established in Procedure 3 excluding those which are constructed by the recursive call of this algorithm is of $O(n)$.

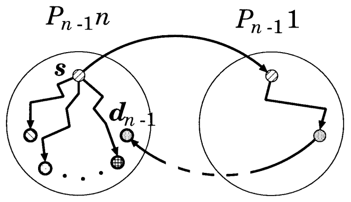


Fig. 9 Case II-2-A.

Proof: The $n-2$ paths selected in step 2 of Procedure 3 are disjoint except for $\boldsymbol{s}$ by the induction hypothesis. The nodes on the paths selected in step 2 have figure n in the last position of their labels. In addition, the nodes on the paths selected in steps 4 to 6 have figures other than n in the last position of their labels, except for $\boldsymbol{s}$ and $\boldsymbol{d}_{n-1}$. Hence, the paths selected in step 2 and the paths selected in steps 4 to 6 are disjoint except for $\boldsymbol{s}$ and $\boldsymbol{d}_{n-1}$. Moreover, the node $\boldsymbol{d}_{n-1}$ does not appear on the paths selected in step 2. Therefore, the $n-1$ paths established in Procedure 3 are disjoint except for $\boldsymbol{s}$.

Moreover, the length of edge selected in step 4 is of $O(1)$. The length of path in step 5 is of $O(n)$. In addition, the length of path constructed in step 6 is of $O(n)$. Summing up them, the sum of the length of paths established in Procedure 3 excluding those which are constructed by the recursive call of the algorithm is of $O(n)$. □

3.4 Case II-2-B-a

In this subsection, we will consider the case which is characterized as follows:

- there is at most one destination node in each class,
- there is no destination node in the class to which the source node $\boldsymbol{s}$ belongs,
- there are some destination nodes which do not belong to the subrotator graph $P_{n-1}n$, and
- at most one destination node exists in a subrotator graph other than $P_{n-1}n$.

By Procedure 4 below, we can construct the $n-1$ paths from $\boldsymbol{s}$ to $n-1$ destination nodes in D which are disjoint except for $\boldsymbol{s}$.

Procedure 4:

1. Let D_1 and D_2 represent the set of destination nodes in $P_{n-1}n$ and other destinations, respectively. Here, we can assume $D_2 = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_k\}$ $(1 \le k \le n-1)$ without loss of generality.
2. Additionally, we can assume without loss of generality that the node $\boldsymbol{d}_k$ has the smallest figure in the last position in its label among the nodes in D_2.
3. Let each destination node $\boldsymbol{d}_i$ belong to the subrotator graph $P_{n-1}l_i$ $(1 \le i \le k-1)$.
4. Select $k-1$ nodes $\boldsymbol{v}_i$ $(1 \le i \le k-1)$ in $P_{n-1}n$ which satisfy following conditions in a greedy manner.

   Condition 1: The first figure of the label of $\boldsymbol{v}_i$ is l_i.
   Condition 2: $\boldsymbol{v}_i \notin D_1$.
5. Obtain $n-2$ paths from $\boldsymbol{s}$ to $D_1 \cup \{\boldsymbol{v}_i | 1 \le i \le k-1\}$ which are disjoint except for $\boldsymbol{s}$ by calling the algorithm recursively.
6. For each $\boldsymbol{v}_i$ $(1 \le i \le k-1)$, select $\boldsymbol{v}_i \rightarrow R_n(\boldsymbol{v}_i)$.

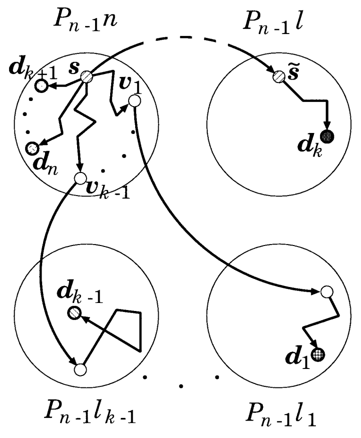


Fig. 10 Case II-2-B-a.

7. For each $R_n(\boldsymbol{v}_i)$ $(1 \le i \le k-1)$, select a shortest path from $R_n(\boldsymbol{v}_i)$ to $\boldsymbol{d}_i$.
8. Let $P_{n-1}l$ be a subrotator graph to which $\boldsymbol{d}_k$ belongs.
9. Let $\tilde{\boldsymbol{s}}$ be a node which is in the class $C(\boldsymbol{s})$ and also belongs to $P_{n-1}l$.
10. Select a path from $\boldsymbol{s}$ to $\tilde{\boldsymbol{s}}$ within the class $C(\boldsymbol{s})$.
11. Establish a shortest path from $\tilde{\boldsymbol{s}}$ to $\boldsymbol{d}_k$. See Fig. 10.

Lemma 5: The $n-1$ paths $\boldsymbol{s} \rightarrow \boldsymbol{d}_i$ $(1 \le i \le n-1)$ selected in Procedure 4 are disjoint except for $\boldsymbol{s}$. Additionally, the sum of the length of paths established in Procedure 4 excluding those which are constructed by the recursive call of this algorithm is of $O(n^2)$.

Proof: The paths selected in step 5 of Procedure 4 are disjoint except for $\boldsymbol{s}$ by the induction hypothesis. All the nodes except for $\boldsymbol{v}_i$ $(1 \le i \le k-1)$ on the paths selected in steps 6 and 7 have the figure l_i in the last positions of their labels. Hence, they are disjoint. The nodes on the paths selected in step 5 have the figure n in the last position of their labels. Moreover, the figures l_i $(1 \le i \le k-1)$ are all different from n. Hence, the paths selected in step 5 and the paths selected in steps 6 and 7 are disjoint except for $\boldsymbol{s}$ and $\boldsymbol{v}_i$ $(1 \le i \le k-1)$. Similarly, the paths selected in steps 10 and 11 and the paths selected in steps 5 to 7 are easily proved to be disjoint except for $\boldsymbol{s}$. Therefore, the $n-1$ paths established in Procedure 4 are disjoint except for $\boldsymbol{s}$.

Additionally, the sum of the length of $k-1$ edges selected in step 6 is of $O(n)$. The sum of the length of $k-1$ shortest paths selected in step 7 is of $O(n^2)$, and the length of path selected in step 10 is of $O(n)$. Finally, the shortest path established in step 11 has of $O(n)$ length. Summing up them, the sum of the length of paths established in Procedure 4 excluding those which are constructed by the recursive call of the algorithm is of $O(n^2)$. □

3.5 Case II-2-B-b

Finally, in this subsection, we will consider the case which is characterized as follows:

- there is at most one destination node in each class,
- there is no destination node in the class to which the source node $\boldsymbol{s}$ belongs,
- there are some destination nodes which do not belong to the subrotator graph $P_{n-1}n$, and
- there exist multiple destination nodes in one or more subrotator graphs other than $P_{n-1}n$.

Procedure 5 below constructs $n-1$ paths from $\boldsymbol{s}$ to $n-1$ destination nodes in D which are disjoint except for the source $\boldsymbol{s}$.

Procedure 5:

1. Select a path from $\boldsymbol{s}$ to the node of a subrotator graph which has multiple destination nodes within the class $C(\boldsymbol{s})$. Let $P_{n-1}l$ and $\tilde{\boldsymbol{s}}$ represent the subrotator graph and the node, respectively. Additionally, let $D_1 = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \cdots, \boldsymbol{d}_k\}$ $(k \leq n-1)$ be the set of destination nodes in $P_{n-1}l$.
2. For each class to which $n-2$ destination nodes in $D - \{\boldsymbol{d}_1\}$ belong, let each $\boldsymbol{u}_i$ $(1 \leq i \leq n-2)$ represent a node which is in the class and belongs to $P_{n-1}l$.
3. In $P_{n-1}l$, construct $n-2$ internally disjoint paths from $\tilde{\boldsymbol{s}}$ to $\boldsymbol{d}_1$ [5]. If each path includes one of $\boldsymbol{u}_i$'s, the subpath from $\tilde{\boldsymbol{s}}$ to $\boldsymbol{d}_2$ is selected, and specification of the nodes $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are exchanged. Otherwise, one of the paths from $\tilde{\boldsymbol{s}}$ to $\boldsymbol{d}_1$ which do not include the nodes $\boldsymbol{u}_i$'s at all is selected.
4. For each of $n-2$ classes to which each node in $D - \{\boldsymbol{d}_1\}$ belongs, let each $\boldsymbol{v}_i$ $(1 \leq i \leq n-2)$ represent the node which is in the class and also belongs to $P_{n-1}n$.
5. Obtain the paths from $\boldsymbol{s}$ to $\boldsymbol{v}_i$ $(1 \leq i \leq n-2)$ which are disjoint except for $\boldsymbol{s}$ by calling the algorithm recursively.
6. Select paths from each $\boldsymbol{v}_i$ $(1 \leq i \leq n-2)$ to the corresponding node in $D - \{\boldsymbol{d}_1\}$ within the class. See Fig. 11.

Lemma 6: The $n-1$ paths $\boldsymbol{s} \to \boldsymbol{d}_i$ $(1 \leq i \leq n-1)$ constructed in Procedure 5 are disjoint except for $\boldsymbol{s}$. Additionally, the sum of the length of paths established in Procedure 5 excluding those which are constructed by the recursive call of this algorithm is of $O(n^2)$.

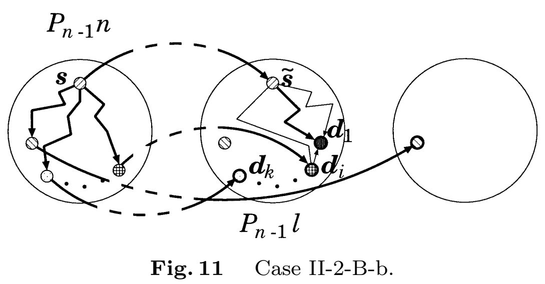


Fig. 11 Case II-2-B-b.

Proof: The paths selected in the step 5 in Procedure 5 are disjoint except for $\boldsymbol{s}$ from the hypothesis of induction. From the property 2 of classes, the paths selected in the step 6 are disjoint. The nodes on the paths selected in the step 5 have the figure n in the last positions of their labels. Additionally, the nodes on the paths selected in the step 6 have other figures than n in the last position of their labels except for $\boldsymbol{v}_i$ $(1 \leq i \leq n-2)$. Hence, the paths selected in the step 5 and the paths selected in the step 6 are disjoint except for $\boldsymbol{s}$ and $\boldsymbol{v}_i$ $(1 \leq i \leq n-2)$. Similarly, the paths selected in the steps 1 and 3 and the paths selected in the steps 5 and 6 are easily proved to be disjoint except for $\boldsymbol{s}$. Therefore, the $n-1$ paths established in Procedure 5 are disjoint except for $\boldsymbol{s}$.

Additionally, the lengths of paths selected in steps 1 and 3 are both of $O(n)$. The sum of the length of $n-2$ paths selected in step 6 is of $O(n^2)$, Summing up them, the sum of the length of paths established in Procedure 5 excluding those which are constructed by the recursive call of the algorithm is of $O(n^2)$. □

Theorem 1: The $n-1$ paths constructed by Procedures from 1 to 5 are disjoint except for $\boldsymbol{s}$. The sum of the length of paths is of $O(n^3)$.

Proof: From Lemmas 2 to 6, the proof is trivial. □

Theorem 2: The complexity of the algorithm is $O(n^5)$.

Proof: We assume that a label of a node is represented by a linear array of n elements. Let $T(n)$ represent the time complexity of our algorithm for an n-rotator graph.

The first case branching operation in our algorithm is begun with calculation of nodes each of which belongs to the same class as each destination node and also belongs to the subrotator graph $P_{n-1}n$. These nodes can be used as representatives of classes to which destination nodes belong. This calculation takes $O(n^3)$ of time complexity. The other tests for other branching can be performed less than $O(n^3)$. Now, the destination nodes are classified into classes and each class has a subset of destination nodes.

In Procedure 1, steps 2 and 4(b)viii are governing and they take $O(n^4)$ of time complexity. The algorithm for the node-to-node disjoint paths problem included in step 4(b)viii is of $O(n^3)$ of time complexity and the sum of the length of paths is of $O(n^2)$ [5]. Hence, the time complexity of Procedure 1 is $T(n) = T(n-1) + O(n^4)$.

Procedure 2 is governed by step 5 which is of $O(n^2)$ of complexity. Therefore, the time complexity of Procedure 2 is $T(n) = T(n-1) + O(n^2)$.

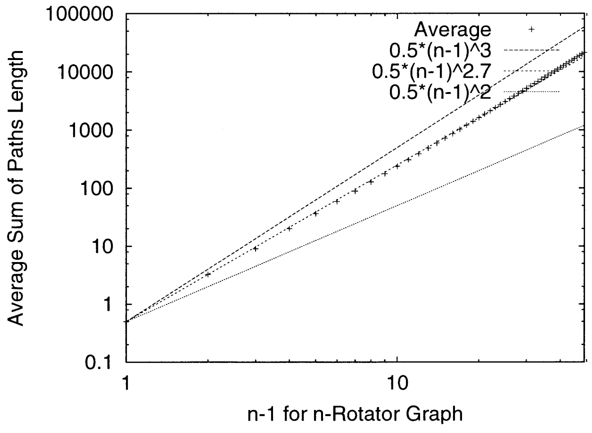


Fig. 12 Average sum of paths length.

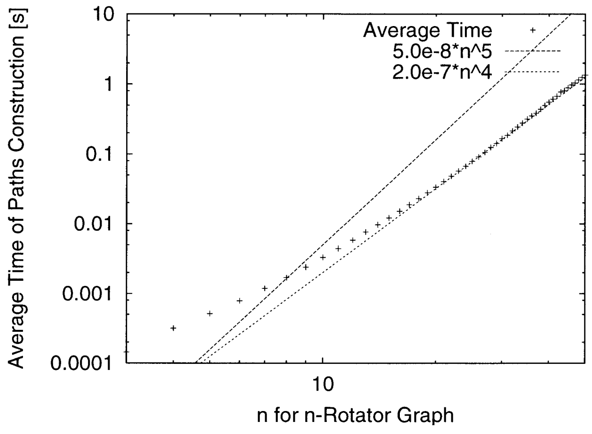


Fig. 13 Average time of paths construction.

In Procedure 3, the check after the recursive call in step 2 is governing. The sum of the length of paths is of $O(n^3)$, and it take $O(n)$ of time to compare two labels, step 3 requires $O(n^4)$ of time complexity. Hence, the complexity of Procedure 3 is $T(n) = T(n-1) + O(n^4)$.

In Procedure 4, step 4 takes $O(n^4)$ of time complexity and it is governing. So, the complexity of Procedure 4 is $T(n) = T(n-1) + O(n^4)$.

Procedure 5 is governed by step 3 which is similar to step 4(b)viii of Procedure 1 and takes $O(n^4)$ of time complexity. Therefore, the time complexity of Procedure 5 is $T(n) = T(n-1) + O(n^4)$.

From above discussion, the time complexity of our algorithm is represented by $T(n) = T(n-1) + O(n^4)$ which results in $T(n) = O(n^5)$. □

4. Simulation

To evaluate the power of the algorithm, we conducted the following computer simulation for an n-rotator graph. The algorithm is implemented by a functional language Haskell to save programming time, and simulation is performed in lazy evaluation manner. The program is compiled by Glasgow Haskell Compiler with `-O2` and `-fglasgow-exts` options and executed on a target machine with an Intel Pentium 700 MHz CPU and a 512 MB memory unit.

1. Select a source node $\boldsymbol{s}$ randomly.
2. Select $n-1$ distinct destination nodes $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, $\cdots$, $\boldsymbol{d}_{n-1}$ randomly.
3. Apply the algorithm and evaluate the sum of the length of paths obtained.

Simulation is performed 1,000 times for each n where $n = 2, \cdots, 50$. Figures 12 and 13 show the average length and time, respectively. From these figures we can see that the average sum of paths length and the average time are of polynomial order and approximately of $O(n^{2.7})$ and of $O(n^4)$ in their ranges.

5. Conclusions

In this paper, we have presented an algorithm for the node-to-set disjoint paths problem in n-rotator graphs which is of polynomial order of n. We also conducted computer simulation to show the average sum of paths being of $O(n^{2.7})$ and the average time being of $O(n^4)$. Future works include detail examination of the algorithm and its improvement.

References

[1] S.B. Akers and B. Krishnamurthy, "A group theoretic model for symmetric interconnection networks," IEEE Trans. Comput., vol.38, no.4, pp.555–566, April 1989.

[2] P.F. Corbett, "Rotator graphs: An efficient topology for point-to-point multiprocessor networks," IEEE Trans. Parallel and Distributed Systems, vol.3, no.5, pp.622–626, May 1992.

[3] K. Ghose and K.R. Desai, "Hierarchical cubic networks," IEEE Trans. Parallel and Distributed Systems, vol.6, no.4, pp.427–435, April 1995.

[4] 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.

[5] 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.

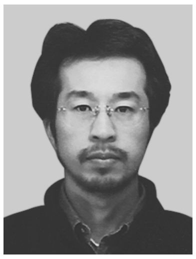
[6] Q.M. Malluhi and M.A. Bayoumi, "The hierarchical hypercube — A new interconnection topology for massively parallel systems," IEEE Trans. Parallel and Distributed Systems, vol.5, no.1, pp.17–30, Jan. 1994.

[7] M.O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," JACM, vol.36, no.2, pp.335–348, Feb. 1989.

[8] P.-J. Wan, "Conflict-free channel set assignment for an optical cluster interconnection network based on rotator digraphs," Theoretical Computer Science, vol.207, no.1, pp.193–201, Jan. 1998.

[9] P.M. Yamakawa, H. Ebara, and H. Nakano, "A routing algorithm in faulty n-rotator graph and its performance evaluation," Trans. IPSJ, vol.56, no.7, pp.1511–1519, July 1995.

Keiichi Kaneko is an Associate Professor at Tokyo University of Agriculture and Technology. 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 also a member of ACM, IPSJ, and JSSST.



Yasuto Suzuki is an undergraduate student at Tokyo University of Agriculture and Technology. His research interests include graph and network theories and fault-tolerant systems.