IEICE TRANS. INF. & SYST., VOL.E86–D, NO.3 MARCH 2003

PAPER

An Algorithm for Node-Disjoint Paths in Pancake Graphs

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

SUMMARY For any pair of distinct nodes in an n-pancake graph, we give an algorithm for construction of $n-1$ internally disjoint paths connecting the nodes in the time complexity of polynomial order of n. The length of each path obtained and the time complexity of the algorithm are estimated theoretically and verified by computer simulation.

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

1. Introduction

In design and implementation of parallel and distributed computing systems, finding disjoint paths in interconnection networks is one of the fundamental issues [9], [12]–[14], [16]. Amongst them is the node-to-node disjoint paths problem: given two nodes $\boldsymbol{s}$ and $\boldsymbol{d}$ in a k-connected graph G, find k internally disjoint paths connecting the nodes. Once this problem is solved, in the communication from $\boldsymbol{s}$ to $\boldsymbol{d}$, node $\boldsymbol{d}$ can receive a message even if the network has $k-1$ faulty elements. This problem can be solved by using the maximum flow technique, which takes polynomial time of the number of nodes. However, if the graph G has many nodes, this approach is not practical.

An n-pancake graph, proposed in [1], is an undirected regular graph with $n!$ nodes and with degree $n-1$. This graph is an attractive alternative to the hypercube for interconnecting processors in a parallel computer because it can interconnect many more processors with a lower degree and a smaller diameter [3]–[5], [7], [15], [17]. In spite of many researches, its accurate diameter and a shortest path algorithm of polynomial time are still unknown. Hence, there is much room for further research. Hence, in this study, we take an n-pancake graph as a target and propose an algorithm to solve the node-to-node disjoint paths problem in the time complexity of polynomial order of n instead of the number of nodes, $n!$.

The rest of this paper is organized as follows. Section 2 introduces some preliminary definitions and a simple routing algorithm. In Sect. 3, our algorithm to obtain node-to-node internally disjoint paths is described and the time complexity and the length of each path are estimated. Then, in Sect. 4, we improve the algorithm to decrease its time complexity. We conduct the computer simulation to verify the time complexity and the length of each path in Sect. 5. Finally, we present our conclusions in Sect. 6.

Manuscript received February 26, 2002.
Manuscript revised September 11, 2002.

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

2. Preliminaries

We first give the definition of a pancake graph [1].

Definition 1: Let $\boldsymbol{u} = u_1u_2\cdots u_n$ be a permutation of n symbols: $1, 2, \cdots, n$. For each i ($1 \leqq i \leqq n$), an operation $\boldsymbol{u}^{(i)}$ is defined as $\boldsymbol{u}^{(i)} = u_iu_{i-1}\cdots u_1u_{i+1}u_{i+2}\cdots u_n$. An undirected graph $G = (V, E)$ is called an n-pancake graph, P_n, if $V = \{u_1u_2\cdots u_n \mid u_1u_2\cdots u_n$ is a permutation of $1, 2, \cdots, n\}$ and $E = \{(\boldsymbol{u}, \boldsymbol{v}) \mid \boldsymbol{u}, \boldsymbol{v} \in V, \boldsymbol{v} = \boldsymbol{u}^{(i)}, 2 \leqq i \leqq n\}$.

Figure 1 presents 2, 3 and 4-pancake graphs. The number of nodes, the degree, the number of edges and the connectivity of an n-pancake graph P_n are $n!$, $n-1$, $(n-1) \times n!/2$ and $n-1$, respectively. Its connectivity can be derived by showing that there is a path between any two nodes even if $n-2$ faulty nodes exist in the graph [2]. Using the algorithm of the node-to-node disjoint paths problem described in this paper, we can

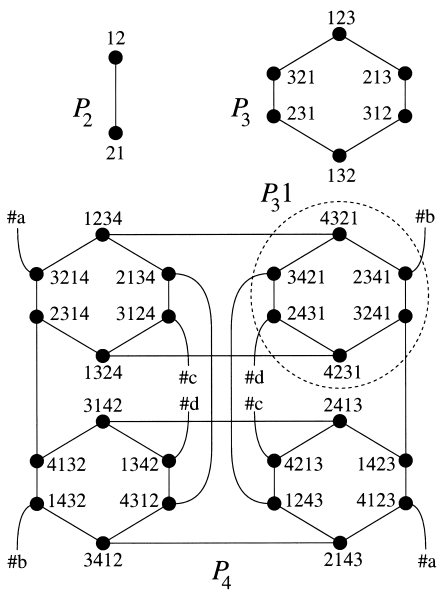


Fig. 1 2, 3 and 4-pancake graphs.

Table 1 Comparison of a pancake graph with other topologies.

	# nodes	degree	diameter
P_n [1], [11]	$n!$	$n-1$	$\leqq 5(n+1)/3$
Q_n [10]	2^n	n	n
T_n [10]	n^2	4	n
S_n [1]	$n!$	$n-1$	$\lfloor \frac{3}{2}(n-1) \rfloor$
R_n [8]	$n!$	$n-1$	$n-1$
$B_{n,k}$ [19]	n^k	$2n$	k
$K_{n,k}$ [6]	$n^k + n^{k-1}$	n	k

obtain such a non-faulty path immediately. However, not vice versa.

P_n contains n different $(n-1)$-subpancake graphs. All nodes in each subpancake graph P_{n-1} share a common last symbol, say k, in their labels. We will specify the subpancake graph by $P_{n-1}k$.

Table 1 shows the comparison of an n-pancake graph P_n with other famous topologies such as an n-cube Q_n, an $n \times n$-torus T_n, an n-star graph S_n, an n-rotator graph R_n, an (n,k)-de Bruijn graph $B_{n,k}$ and an (n,k)-Kautz graph $K_{n,k}$. From this table, we can observe that the n-pancake graph shows a better performance against the topologies Q_n and T_n in that it can connect more nodes for a given degree or a given diameter. The clear comparison with S_n is difficult, but it is known that the diameter of P_n is smaller than that of S_n for any $n < 10$ [11]. P_n is inferior to R_n, $B_{k,n}$ and $K_{k,n}$. However R_n is directed, and $B_{k,n}$ and $K_{k,n}$ neither have symmetry nor recursive structure. Thus they are impractical for running applications.

Next, we define the terminology *internally disjoint* and the node-to-node disjoint paths problem.

Definition 2: Two paths connecting two nodes are said to be internally disjoint iff two paths share no nodes except for both ends. A set of paths connecting two nodes is said to be internally disjoint iff any pair of paths in the set are internally disjoint.

Definition 3: The node-to-node disjoint paths problem in a k-connected graph is to find k paths from a source node $\boldsymbol{s}$ to a destination node $\boldsymbol{d}$ which are internally disjoint.

For Q_n, S_n and R_n, algorithms of polynomial time of n for this problem have already been developed [9], [13], [18].

As mentioned above, an algorithm to find a shortest path in an n-pancake graph in polynomial time of n has not been discovered yet. Therefore, in this paper, we use the algorithm `Route` shown in Fig. 2 for routing between two nodes where $\boldsymbol{u}^{(i_1,i_2,\cdots,i_k)}$ denotes the node obtained by $(\boldsymbol{u}^{(i_1,i_2,\cdots,i_{k-1})})^{(i_k)}$ [1]. Note that $\boldsymbol{u}^{(1)} = \boldsymbol{u}^{(i,i)} = \boldsymbol{u}$.

Assuming that each node is represented by an array and we can store a number less than or equal to n in one word, we observe following lemma about the algorithm.

Lemma 1: `Route` gives an elementary path between two arbitrary nodes $\boldsymbol{s}$ and $\boldsymbol{d}$ in an n-pancake graph. The length of this path and the time complexity of `Route` are of $O(n)$ and $O(n^2)$, respectively.

```
procedure Route(s=s_1 s_2 ... s_n, d=d_1 d_2 ... d_n);
begin
  for i := n to 1 step -1
    if s_i ≠ d_i then begin
      find k such that s_k = d_i;
      select an edge (s, s^(k));
      select an edge (s^(k), s^(k,i));
      s := s^(k,i);
    end
end;
```

Fig. 2 A polynomial-time routing algorithm.

3. Algorithm

In this section, we give an algorithm for the node-to-node disjoint paths problem in P_n.

Because of the symmetry of P_n, we can fix the source node $\boldsymbol{s}$ to be $12\cdots n$ without any loss of generality. For a 2-pancake graph P_2, the problem is trivial and we assume that $n > 2$ in the following. Let the destination node $\boldsymbol{d} = d_1 d_2 \cdots d_n$. In the permutation $\boldsymbol{d}$, we assume that each symbol i is at the $p(i)$th position (i.e. $d_{p(i)} = i$).

Our algorithm is divided into three procedures according to the cases given below:

Case I $\boldsymbol{d} = d_1 d_2 \cdots d_{n-1} n$,

Case II $\boldsymbol{d} = n d_2 d_3 \cdots d_n$,

Case III Otherwise.

Procedures to construct $n-1$ paths $r_1, r_2, \cdots, r_{n-1}$ for these three cases are described below.

Case I $\boldsymbol{d} = d_1 d_2 \cdots d_{n-1} n$

Construction of r_i $(2 \leqq i \leqq n-1)$**:**

Obtain $n-2$ paths from $\boldsymbol{s}$ to $\boldsymbol{d}$ which are internally disjoint by calling the algorithm recursively inside the $P_{n-1}n$.

Construction of r_1**:**

1. Select a path $\boldsymbol{s} \rightarrow \boldsymbol{s}^{(n)} \rightarrow \boldsymbol{s}^{(n,n-d_1+1)} \rightarrow \boldsymbol{s}^{(n,n-d_1+1,n)}$.
2. In $P_{n-1}d_1$, by calling `Route`, select a polynomial path from $\boldsymbol{s}^{(n,n-d_1+1,n)}$ to $\boldsymbol{d}^{(n)}$.
3. Select an edge $\boldsymbol{d}^{(n)} \rightarrow \boldsymbol{d}$ (see Fig. 3).

Lemma 2: The paths r_i $(1 \leqq i \leqq n-1)$ constructed in Case I are internally disjoint.

Proof: The $n-2$ paths r_i $(2 \leqq i \leqq n-1)$ are internally disjoint by the induction hypothesis. All nodes on these paths have the same last symbol n in their labels. On the other hand, except for $\boldsymbol{s}$ and $\boldsymbol{d}$, all nodes on the r_1 do not have the symbol n at the last positions. Hence, r_i $(1 \leqq i \leqq n-1)$ are internally disjoint. □

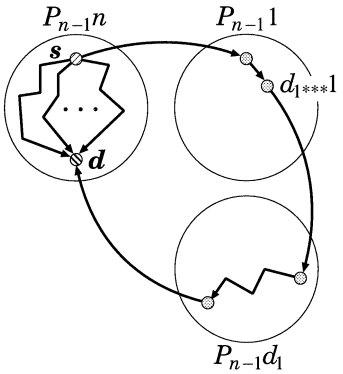


Fig. 3 Case I ($\boldsymbol{d} = d_1 d_2 \cdots d_{n-1} n$).

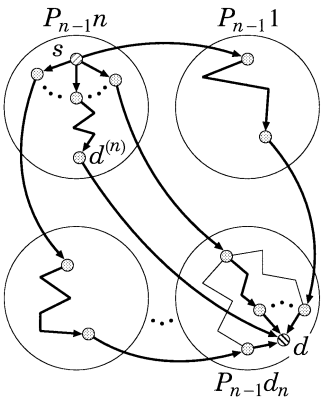


Fig. 4 Case II ($\boldsymbol{d} = n d_2 d_3 \cdots d_n, d_n \neq x$).

Case II $\boldsymbol{d} = n d_2 d_3 \cdots d_n$

First, construct a polynomial path from $\boldsymbol{s}$ to $\boldsymbol{d}^{(n)}$ in $P_{n-1}n$. If the path includes one or some of $\boldsymbol{s}^{(i)}$'s ($2 \leqq i \leqq n-1$), select the nearest node $\boldsymbol{s}^{(x)}$ from $\boldsymbol{d}^{(n)}$. Otherwise, that is, if $\boldsymbol{d} = n(n-1)\cdots 1$, let x be $d_n = 1$.

Construction of r_i ($1 \leqq i \leqq n-1,\ i \neq x,\ i \neq d_n$)**:**

1. From $\boldsymbol{s}$, select a path $\boldsymbol{s} \to \boldsymbol{s}^{(i)} \to \boldsymbol{s}^{(i,n)}$.
2. In $P_{n-1}i$, by calling `Route`, select a polynomial path from $\boldsymbol{s}^{(i,n)}$ to $\boldsymbol{d}^{(p(i),n)}$.
3. Select a path $\boldsymbol{d}^{(p(i),n)} \to \boldsymbol{d}^{(p(i))} \to \boldsymbol{d}$.

Construction of r_x**:**

1. Select an edge $\boldsymbol{s} \to \boldsymbol{s}^{(x)}$.
2. Select the subpath from $\boldsymbol{s}^{(x)}$ to $\boldsymbol{d}^{(n)}$ of the polynomial path constructed at the beginning.
3. Select an edge $\boldsymbol{d}^{(n)} \to \boldsymbol{d}$.

Construction of r_{d_n} **(in case that** $d_n \neq x$**):**

1. Select a path $\boldsymbol{s} \to \boldsymbol{s}^{(d_n)} \to \boldsymbol{s}^{(d_n,n)}$.
2. In $P_{n-1}d_n$, obtain $n-2$ paths from $\boldsymbol{s}^{(d_n,n)}$ to $\boldsymbol{d}$ which are internally disjoint by calling the algorithm recursively. Among them, select the path that has $\boldsymbol{d}^{(p(x))}$ on it (see Fig. 4).

Lemma 3: The paths r_i ($1 \leqq i \leqq n-1$) constructed in Case II are internally disjoint.

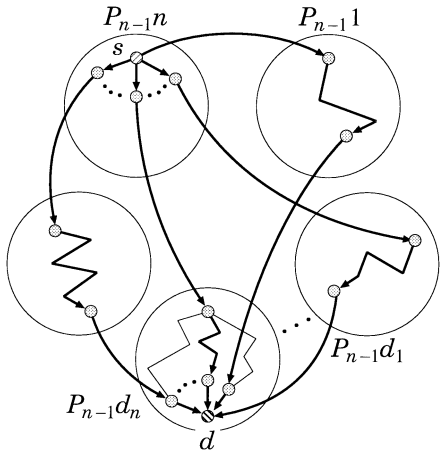


Fig. 5 Case III.

Proof: Except for $\boldsymbol{s}$ and $\boldsymbol{d}$, all nodes on r_i ($1 \leqq i \leqq n-1,\ i \neq x,\ i \neq d_n$) satisfy one of the following two conditions.

1. The first symbol is i, and the last symbol is either n or d_n.
2. The last symbol is i.

Hence, r_i ($1 \leqq i \leqq n-1,\ i \neq x,\ i \neq d_n$) are internally disjoint if i's are different.

Similarly, it can be proved that r_x, r_{d_n} and r_i ($1 \leqq i \leqq n-1,\ i \neq x,\ i \neq d_n$) are internally disjoint. □

Case III Otherwise

Construction of r_i ($1 \leqq i \leqq n-1,\ i \neq d_n$)**:**

1. Select a path $\boldsymbol{s} \to \boldsymbol{s}^{(i)} \to \boldsymbol{s}^{(i,n)}$.
2. In $P_{n-1}i$, by calling `Route`, select a polynomial path from $\boldsymbol{s}^{(i,n)}$ to $\boldsymbol{d}^{(p(i),n)}$.
3. From $\boldsymbol{d}^{(p(i),n)}$, select a path $\boldsymbol{d}^{(p(i),n)} \to \boldsymbol{d}^{(p(i))} \to \boldsymbol{d}$.

Construction of r_{d_n}**:**

1. Select a path $\boldsymbol{s} \to \boldsymbol{s}^{(d_n)} \to \boldsymbol{s}^{(d_n,n)}$.
2. In $P_{n-1}d_n$, obtain $n-2$ paths from $\boldsymbol{s}^{(d_n,n)}$ to $\boldsymbol{d}$ which are internally disjoint by calling the algorithm recursively. Among them, select the path that has $\boldsymbol{d}^{(p(n))}$ on it (see Fig. 5).

Lemma 4: The paths r_i ($1 \leqq i \leqq n-1$) constructed in Case III are internally disjoint.

Proof: Similar to Case II, the $n-1$ paths r_i ($1 \leqq i \leqq n-1$) are internally disjoint. □

From Lemma 1 to 3, the following theorem holds immediately.

Theorem 1: The $n-1$ paths constructed by our algorithm are internally disjoint.

In addition, we can prove the following theorems. We assume that a label of a node is represented by a linear array of n elements.

Theorem 2: The length of each path obtained by our algorithm for P_n is $O(n)$.

Proof: Let $l(n)$ represent the maximum path length by our algorithm for P_n.

Then, $l(n)$ for Cases I, II and III are $\max\{l(n-1), O(n)\}$, $\max\{l(n-1)+2, O(n)\}$ and $\max\{l(n-1)+2, O(n)\}$, respectively. From this observation, we obtain $l(n) = O(n)$ immediately. □

Theorem 3: The complexity of the algorithm is $O(n^4)$.

Proof: Let $T(n)$ represent the time complexity of our algorithm for P_n.

The first case branching operation in our algorithm can be done in $O(1)$ time.

In Case III, r_i $(1 \leqq i \leqq n-1,\ i \neq d_n)$ are constructed in $O(n^3)$ time. In addition, r_{d_n} is constructed in $T(n-1)+O(n^2)$ time. Hence, the time complexity of Case III is of $T(n-1)+O(n^3)$.

Similarly, the time complexities of Case I and Case II are of $T(n-1)+O(n^2)$ and of $T(n-1)+O(n^3)$, respectively.

From above discussion, the time complexity of our algorithm is represented by $T(n) = T(n-1)+O(n^3)$, which results in $T(n) = O(n^4)$. □

4. Improvement of the Algorithm

In the algorithm described above, constructions of r_{d_n} of Case II and Case III call the algorithm recursively. These recursive calls are used to obtain the paths from $\boldsymbol{s}^{(d_n,n)}$ to $\boldsymbol{d}$ in $P_{n-1}d_n$ that include the particular neighbor nodes of $\boldsymbol{d}$. If such paths can be obtained in $O(n^3)$ time, the total complexity of the algorithm decreases to $O(n^3)$.

In this section, we show an algorithm to obtain such paths in $O(n^2)$ time.

4.1 Formalization of the Problem

Since an n-pancake graph is undirected, we could formalize the problem as below.

Definition 4: The constrained path problem in an n-pancake graph is to find a path from a source node $\boldsymbol{s}$ to a destination node $\boldsymbol{d}$ that includes a particular neighbor node $\boldsymbol{t}$ of $\boldsymbol{s}$ and does not include any other neighbor nodes.

4.2 Algorithm

In this subsection, we give an algorithm for the constrained path problem in P_n.

Let $\boldsymbol{s} = 12\cdots n$, $\boldsymbol{d} = d_1 d_2 \cdots d_n$ and $\boldsymbol{t} = t_1 t_2 \cdots t_n$. For P_3, the problem is trivial and we assume that $n > 4$ in the following.

Case I' $\boldsymbol{d} \in P_{n-1}n$

1. If $\boldsymbol{t} \in P_{n-1}n$, then
   In $P_{n-1}n$, obtain the constrained path from $\boldsymbol{s}$ to $\boldsymbol{d}$ by calling the algorithm recursively.

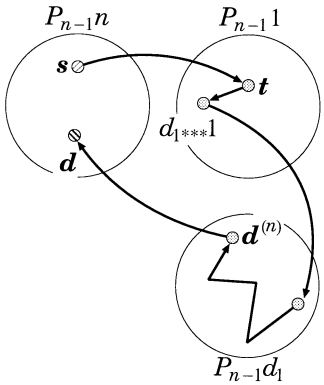


Fig. 6 Constrained path between $\boldsymbol{s}$ and $\boldsymbol{d}$ (1).

2. If $\boldsymbol{t} \in P_{n-1}1$ and $d_1 = 1$, then
   a. Select an edge $\boldsymbol{s} \to \boldsymbol{t}$
   b. In $P_{n-1}1$, by calling `Route`, select a polynomial path from $\boldsymbol{t}$ to $\boldsymbol{d}^{(n)}$.
   c. Select an edge $\boldsymbol{d}^{(n)} \to \boldsymbol{d}$.
3. If $\boldsymbol{t} \in P_{n-1}1$ and $d_1 \neq 1$, then
   a. Assume that $t_k = d_1$, that is, $p(t_k) = 1$.
   b. Select a path $\boldsymbol{s} \to \boldsymbol{t} \to \boldsymbol{t}^{(k)} \to \boldsymbol{t}^{(k,n)}$.
   c. In $P_{n-1}d_1$, by calling `Route`, select a polynomial path from $\boldsymbol{t}^{(k,n)}$ to $\boldsymbol{d}^{(n)}$.
   d. Select an edge $\boldsymbol{d}^{(n)} \to \boldsymbol{d}$ (see Fig. 6).

Case II' $\boldsymbol{d} \notin P_{n-1}n$

1. If $\boldsymbol{t} \in P_{n-1}n$ and $\boldsymbol{d} \in P_{n-1}t_1$, then
   a. Select a path $\boldsymbol{s} \to \boldsymbol{t} \to \boldsymbol{t}^{(n)}$.
   b. In $P_{n-1}t_1$, by calling `Route`, select a polynomial path from $\boldsymbol{t}^{(n)}$ to $\boldsymbol{d}$.
2. If $\boldsymbol{t} \in P_{n-1}n$ and $\boldsymbol{d} \notin P_{n-1}t_1$, then
   a. Select a path $\boldsymbol{s} \to \boldsymbol{t} \to \boldsymbol{t}^{(n)}$.
   b. If $t_1 = d_1$, then
      i. In $P_{n-1}t_1$, by calling `Route`, select a polynomial path from $\boldsymbol{t}^{(n)}$ to $\boldsymbol{d}^{(n)}$.
      ii. Select an edge $\boldsymbol{d}^{(n)} \to \boldsymbol{d}$.
   c. If $t_1 \neq d_1$, then
      i. In $P_{n-1}t_1$, by calling `Route`, select a polynomial path from $\boldsymbol{t}^{(n)}$ to $\boldsymbol{d}^{(p(t_1),n)}$.
      ii. Select a path $\boldsymbol{d}^{(p(t_1),n)} \to \boldsymbol{d}^{(p(t_1))} \to \boldsymbol{d}$ (see Fig. 7).
3. If $\boldsymbol{t} \in P_{n-1}1$ and $\boldsymbol{d} \in P_{n-1}1$, then
   a. Select an edge $\boldsymbol{s} \to \boldsymbol{t}$.
   b. In $P_{n-1}1$, by calling `Route`, select a polynomial path from $\boldsymbol{t}$ to $\boldsymbol{d}$.
4. If $\boldsymbol{t} \in P_{n-1}1$ and $\boldsymbol{d} \notin P_{n-1}1$, then
   a. Assume that $t_k = d_n$, that is, $p(t_k) = n$.
   b. Select a path $\boldsymbol{s} \to \boldsymbol{t} \to \boldsymbol{t}^{(k)} \to \boldsymbol{t}^{(k,n)}$.

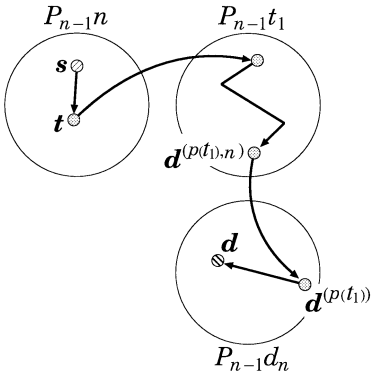


Fig. 7 Constrained path between s and d (2).

c. In $P_{n-1}d_n$, by calling `Route`, select a polynomial path from $t^{(k,n)}$ to d.

Lemma 5: The paths obtained by Case I' and II' include t only as the neighbor of s. Its complexities are $O(n^2)$ both, and the length of the paths are $O(n)$ both.

Proof: In Case I':

If $t \in P_{n-1}n$, the obtained path satisfies the constraint by the assumption. Excluding the recursive call, its time complexity is of $O(1)$.

If $t \in P_{n-1}1$ and $d_1 = 1$, both t and $d^{(n)}$ are in $P_{n-1}1$. Hence, except for t, the obtained path does not include the neighbor of s. Its time complexity is $O(n^2)$.

If $t \in P_{n-1}1$ and $d_1 \neq 1$, node $t^{(k,n)}$ does not belong to $P_{n-1}1$ nor $P_{n-1}n$, but to $P_{n-1}d_1$. Then, the path satisfies the constraint. Its time complexity is $O(n^2)$.

In addition, the length of the path obtained in the case is $O(n)$.

In Case II', the similar discussion holds. □

From Lemma 4 and proofs of Theorems 2 and 3, we can derive the following theorem.

Theorem 4: The $n-1$ paths constructed by the improved algorithm are internally disjoint. The length of each path and the time complexity of the algorithm are $O(n)$ and $O(n^3)$, respectively.

5. Simulation

To evaluate the performance of the algorithm, we conducted the following computer simulation for an n-pancake graph. The algorithm is implemented by a programming language C. The program is compiled by `gcc` with `-O2` option and executed on a target machine with an Intel PentiumIII 700 MHz CPU and a 128 MB memory unit.

1. Fix a source node s to be $12\cdots n$.
2. Select a destination node d other than s randomly.
3. Apply the improved algorithm and measure the sum of the lengths of paths obtained and execution time.

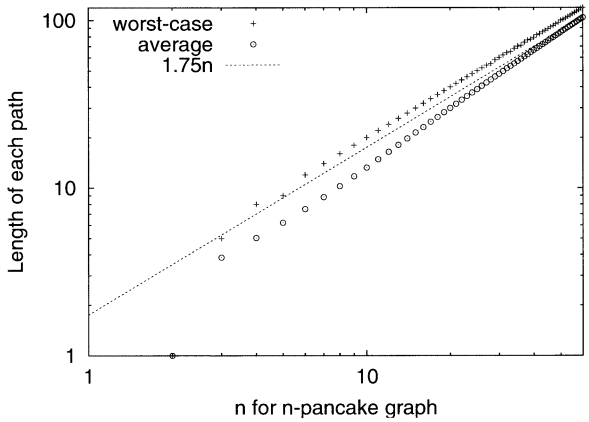


Fig. 8 Length of each path.

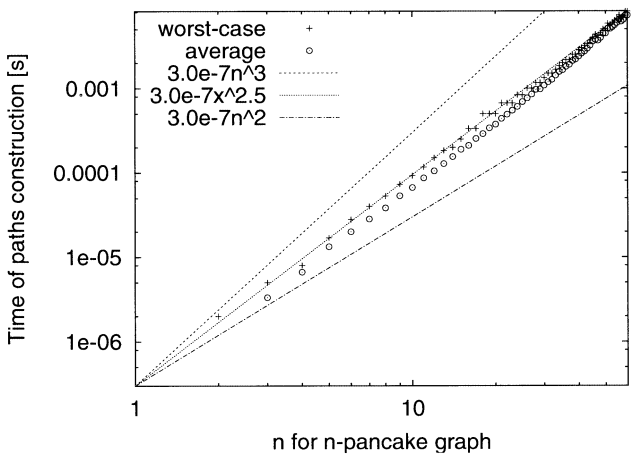


Fig. 9 Time of paths construction.

Simulation is performed 10,000 times for each n where $n = 2, 3, \cdots, 60$. Results are shown in Figs. 8 and 9. From these figures we can observe that the average length of each path and the average time of paths construction are of polynomial order and approximately $O(n)$ and $O(n^{2.5})$ in their ranges.

6. Conclusions

In this paper, we have presented an $O(n^3)$-time algorithm for the node-to-node disjoint paths problem in an n-pancake graph. The length of each path is $O(n)$. We also conducted the computer simulation to show the average time being $O(n^{2.5})$ and the average length of each path being $O(n)$. It would also be interesting to find efficient algorithms for the node-to-node disjoint paths problem in other interconnection networks.

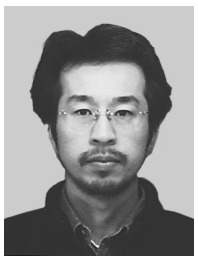
Acknowledgement

This work was partially supported by Atoda Foundation of Division of Electronic and Information Engineering, Graduate School of Technology, Tokyo University

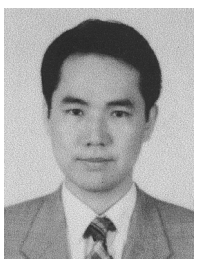
of Agriculture and Technology, and by Grant-in-Aid for Scientific Research (C) of Japan Society for the Promotion of Science under Grant No. 13680398.

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] S.B. Akers and B. Krishnamurthy, "Group graphs as interconnection networks," Proc. 14th International Conference on Fault Tolerant Computing, pp.422–427, 1984.
[3] S.G. Akl and K. Qiu, "A novel routing scheme on the star and pancake interconnection networks and its applications," Parallel Computing, vol.19, no.1, pp.95–101, Jan. 1993.
[4] S.G. Akl and K. Qiu, "Parallel minimum spanning forest algorithms on the star and pancake interconnection networks," Proc. Joint Conference Vector & Parallel Processing, pp.565–570, 1993.
[5] S.G. Akl, K. Qiu, and I. Stojmenović, "Fundamental algorithms for the star and pancake interconnection networks with applications to computational geometry," Networks, vol.23, no.4, pp.215–226, April 1993.
[6] J-C. Bermond, R. Dawes, and F.O. Ergincan, "De Bruijn and Kautz bus networks," Networks, vol.30, no.3, pp.205–218, Oct. 1997.
[7] P. Berthomé, A Ferreira, and S. Perennes, "Optimal information dissemination in star and pancake networks," IEEE Trans. Parallel Distrib. Syst., vol.7, no.12, pp.1292–1300, July 1996.
[8] P.F. Corbett, "Rotator graphs: An efficient topology for point-to-point multiprocessor networks," IEEE Trans. Parallel Distrib. Syst., vol.3, no.5, pp.622–626, March 1992.
[9] M. Dietzfelbinger, S. Madhavapeddy, and I.H. Sudborough, "Three disjoint path paradigms in star networks," Proc. IEEE SPDP, pp.400–406, 1991.
[10] S.J. Duato, S. Yalamanchili, and L. Ni, Interconnection Networks: An Engineering Approach, IEEE Computer Society Press, 1997.
[11] W.H. Gates and C.H. Papadimitriou, "Bounds for sorting by prefix reversal," Discrete Mathematics, vol.27, no.6, pp.47–57, June 1979.
[12] Q. Gu and S. Peng, "Node-to-set disjoint paths problem in star graphs," Inf. Process. Lett., vol.62, no.4, pp.201–207, April 1997.
[13] 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.
[14] 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.
[15] A. Kanevsky and C. Feng, "On the embedding of cycles in pancake graphs," Parallel Computing, vol.21, no.6, pp.923–936, June 1995.
[16] S. Madhavapeddy and I.H. Sudborough, "A topological property of hypercubes: Node disjoint paths," Proc. IEEE SPDP, pp.532–539, 1990.
[17] K. Qiu, H. Meijer, and S.G. Akl, "Parallel routing and sorting on the pancake network," Proc. International Conference on Computing and Information, pp.360–371, 1991.
[18] Y. Saad and M.H. Schultz, "Topological properties of hypercubes," IEEE Trans. Comput., vol.37, no.7, pp.867–872, July 1988.
[19] M.R. Samatham and D.K. Pradhan, "The de Bruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI," IEEE Trans. Comput., vol.38, no.4, pp.567–581, April 1989.



Yasuto Suzuki received the B.E. degree from Tokyo University of Agriculture and Technology in 2001. His research interests include graph and network theories and fault-tolerant systems.



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.