Genome rearrangements: efficient algorithm in pancake graphs with one faulty node

Jer-Shyan Wu, Yu-Song Hou and Wei-Qian Cai
Department of Bioinformatics
Chung Hua University, 30012 Hsinchu, Taiwan
{jswu,yshou,m10120001}@chu.edu.tw

Abstract

Sorting by prefix reversals on genome rearrangements, pancake graph can be constructed. For node-to-node routing on n-pancake graph, Gates and Papadimitriou first proposed O(n) algorithm with at most 5(n + 1) / 3 length of the path in 1979. If there exist (n - 2) faulty nodes, Kaneko and Sawada proposed O(n) algorithm with at most 5(n + 1) / 3 + (8 / 3) length of the path in 2007; extending to (n - 2) faulty clusters with diameter 2, they also proposed O(n) algorithm with at most 5(n + 1) / 3 + 7 length of the path.

Consider the well-designed steady system, the probability of faulty node is very low, and it can be repaired well soon. So the most case of fault-tolerant node-to-node routing is only one faulty node. In this paper, we propose an efficient fault-tolerant node-to-node routing algorithm in n-pancake graph with one faulty node, the minimum length of the routing path is at most (5n / 3) + 3 = 5(n + 1) / 3 + (4 / 3), and the time complexity is still O(n).

1 Introduction

Genome rearrangements are given two distinct objects with the same genomes but different ordering to transfer each other. Sorting by prefix reversals is very important transfer operation, and pancake graph can be constructed.

An n-pancake graph denoted as P_n is an undirected graph with n! nodes and n – 1 links per node[1]. Gates and Papadimitriu[2] first proposed routing algorithm from any pair of nodes, $d(P_n)$ denoted as the minimum length of the node-to-node routing path is at most 5(n + 1) / 3, and the time complexity is O(n). Consider P_n with n - 2 faulty nodes, Kaneko, Sawada and Peng[3] designed an O(n) fault-tolerant node-to-node routing algorithm with at most $d(P_n)$ + (8 / 3) length. Extending to n - 2 faulty clusters whose diameter are at most 2, and then they also proposed O(n) fault-tolerant node-to-node routing algorithm with at most $d(P_n)$ + 7 length.

Consider the well-designed steady system, the probability of faulty node is very low, and it can be repaired well soon. So the most case of fault-tolerant node-to-node routing is only one faulty node. In this paper, we propose an efficient fault-tolerant node-to-node routing algorithm in n-pancake graph with one faulty node, the minimum length of the routing path is at most (5n / 3) + 3 = 5(n + 1) / 3 + (4 / 3) = $d(P_n)$ + (4 / 3), and the time complexity is still O(n).

2 Notation

We first introduce some definitions and theorems to explain the prefix reversal, n-pancake graph, and sub n-pancake graph.

Definition 1: Given integer n, and let $u = (u_1, u_2, \cdots, u_{i-1}, u_i, u_{i+1}, \cdots, u_n)$ be a permutation of $[n] = \{1, 2, \cdots, n\}$. The prefix reversal is defined $u^{(i)} = (u_i, u_{i-1}, \cdots, u_2, u_1, u_{i+1}, \cdots, u_n)$, for i = 2, 3, ⋯, n. $u^{(i, j)}$ is defined as continuous prefix reversals with index i and j.

Definition 2: An undirected graph G = (V, E) is called an n-pancake graph denoted as P_n, where $V = \{u|\ u$ is a permutation of $[n]\}$, and $E = \{(u, u^{(i)})|\ u \in V, 2 \le i \le n\}$.

Definition 3: The sub n-pancake graph G = (V, E) induced by P_n, is denoted as $P_n(k)$, where $V = \{(u_1, u_2, \cdots, u_{n-1}, k)|\ (u_1, u_2, \cdots, u_{n-1})$ is a permutation of $[n]\backslash\{k\}\}$, and $E = \{(u, u^{(i)})|\ u \in V, 2 \le i \le n - 1\}$, for $1 \le k \le n$.

Theorem 1: (n - 1)-pancake graph P_{n-1} is isomorphic to sub n-pancake graph $P_n(k)$.

Proof: Give an (n - 1)-pancake graph P_{n-1}, let $u = (u_1, u_2, \cdots, u_{i-1}, u_i, u_{i+1}, \cdots, u_{n-1})$ be the selected node of P_{n-1}. If $u_i = k$, then the mapping node to $P_n(k)$ is $(u_1, u_2, \cdots, u_{i-1}, n, u_{i+1}, \cdots, u_{n-1}, k)$, for $1 \le k \le n$. An n-pancake graph P_n can be recursively constructed by n disjoint (n-1)-pancake graphs mapping to $P_n(1), P_n(2), \cdots, P_n(n)$.

Figure 1 show examples of pancake graphs P_3 and P_4, there are four sub pancake graphs: $P_4(1)$, $P_4(2)$, $P_4(3)$, and $P_4(4)$ among P_4.

Theorem 2: Given an n-pancake graph P_n, consider the node-to-node routing algorithm, $d(P_n)$ denoted as the minimum length of the routing path is at most 5(n + 1)/3, and the time complexity is

O(n).

Proof: The algorithm is proposed and proved by Gates and Papadimitriu[2].

3 Algorithms

Given an n-pancake graph P_n, and the source node $s = (s_1, s_2, \cdots, s_n)$, the destination node $t = (t_1, t_2, \cdots, t_n)$, the faulty node $f = (f_1, f_2, \cdots, f_n)$. We propose an efficient fault-tolerant routing algorithm from the source node s to the destination node s to avoid the faulty node f.

Figure 1-5 are the details of the routing algorithm, there are total 11 cases. They are explained as follows.

Case 1: $t \in P_n(s_n)$, $f \in P_n(s_n)$, $t_1 = s_1$.

Nodes s, t, f, are in $P_n(s_n)$, we first do prefix reversals $s^{(n)}$ and $t^{(n)}$, now these two derived nodes are in $P_n(t_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 2 = 5[(n - 1) + 1] / 3 + 2 = (5n / 3) + 2 = 5(n + 1) / 3 + (1 / 3) = d(P_n) + (1 / 3)$, and the time complexity is $O(n - 1) + 2 = O(n)$.

Case 2: $t \in P_n(s_n)$, $f \in P_n(s_n)$, $t_1 \neq s_1$, $t_1 = f_1$.

Nodes s, t, f, are in $P_n(s_n)$, we first do prefix reversal $s^{(n)}$. If $t_k = s_1$, then we do $t^{(k, n)}$. Now these two derived nodes are in $P_n(s_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 3 = 5[(n - 1) + 1] / 3 + 3 = (5n / 3) + 3 = 5(n + 1) / 3 + (4 / 3) = d(P_n) + (4 / 3)$, and the time complexity is $O(n - 1) + 3 = O(n)$.

Case 3: $t \in P_n(s_n)$, $f \in P_n(s_n)$, $t_1 \neq s_1$, $t_1 \neq f_1$.

Nodes s, t, f, are in $P_n(s_n)$, we first do prefix reversal $t^{(n)}$. If $t_1 = s_k$, then we do $s^{(k, n)}$. Now these two derived nodes are in $P_n(t_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 3 = 5[(n - 1) + 1] / 3 + 3 = (5n / 3) + 3 = 5(n + 1)/3 + (4 / 3) = d(P_n) + (4 / 3)$, and the time complexity is $O(n - 1) + 3 = O(n)$.

Case 4: $t \in P_n(s_n)$, $f \notin\in P_n(s_n)$.

Nodes s and t are in $P_n(s_n)$, we directly implement the routing algorithm proposed by Gates and Papadimitriu[2] from the source nodes s to the destination node t.

The minimum length of the routing path is at most $d(P_{n-1}) = 5[(n - 1) + 1] / 3 = 5n / 3 = 5(n + 1) / 3 - (5 / 3) = d(P_n) - (5 / 3)$, and the time complexity is $O(n - 1) = O(n)$.

Case 5: $t \notin P_n(s_n)$, $f \in P_n(t_n)$, $t_1 = s_n$.

We first do prefix reversal $t^{(n)}$, the derived node and s are in $P_n(s_n)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 1 = 5[(n - 1) + 1] / 3 + 1 = (5n / 3) + 1 = 5(n + 1) / 3 - (2 / 3) = d(P_n) - (2 / 3)$, and the time complexity is $O(n - 1) + 1 = O(n)$.

Case 6: $t \notin P_n(s_n)$, $f \in P_n(t_n)$, $t_1 \neq s_n$, $t_1 = s_1$.

We first do prefix reversals $s^{(n)}$ and $t^{(n)}$, these two derived nodes are in $P_n(t_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 2 = 5[(n - 1) + 1] / 3 + 2 = (5n / 3) + 2 = 5(n + 1) / 3 + (1 / 3) = d(P_n) + (1 / 3)$, and the time complexity is $O(n - 1) + 2 = O(n)$.

Case 7: $t \notin P_n(s_n)$, $f \in P_n(t_n)$, $t_1 \neq s_n$, $t_1 \neq s_1$.

We first do prefix reversal $t^{(n)}$. If $s_k = t_1$, then we do $s^{(k, n)}$. Now these two derived nodes are in $P_n(t_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 3 = 5[(n - 1) + 1] / 3 + 3 = (5n / 3) + 3 = 5(n + 1) / 3 + (4 / 3) = d(P_n) + (4 / 3)$, and the time complexity is $O(n - 1) + 3 = O(n)$.

Case 8: $t \notin P_n(s_n)$, $f \notin P_n(t_n)$, $f \in P_n(s_n)$, $t_1 = s_1$.

We first do prefix reversals $s^{(n)}$ and $t^{(n)}$, these two derived nodes are in $P_n(s_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 2 = 5[(n - 1) + 1] / 3 + 2 = 5n / 3 + 2 = 5(n + 1) / 3 + (1 / 3) = d(P_n) + (1 / 3)$, and the time complexity is $O(n - 1) + 2 = O(n)$.

Case 9: $t \notin P_n(s_n)$, $f \notin P_n(t_n)$, $f \in P_n(s_n)$, $t_1 \neq s_1$.

We first do prefix reversal $s^{(n)}$. If $t_k = s_1$, then we do $t^{(k, n)}$. Now these two derived nodes are in $P_n(s_1)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 3 = 5[(n - 1) + 1] / 3 + 3 = 5n / 3 + 3 = 5(n + 1) / 3 + (4 / 3) = d(P_n) + (4 / 3)$, and the time complexity is $O(n - 1) + 3 = O(n)$.

Case 10: $t \notin P_n(s_n)$, $f \notin P_n(t_n)$, $f \notin P_n(s_n)$, $t_1 = s_n$.

We first do prefix reversal $t^{(n)}$, the derived node and s are in $P_n(s_n)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 1 = 5[(n - 1) + 1] / 3 + 1 = (5n / 3) + 1 = 5(n + 1) / 3 - (2 / 3) = d(P_n) - (2 / 3)$, and the time complexity is $O(n - 1) + 1 = O(n)$.

Case 11: $t \notin P_n(s_n)$, $f \notin P_n(t_n)$, $f \notin P_n(s_n)$, $t_1 \neq s_n$.

If $t_k = s_n$, then we do $t^{(k, n)}$. Now the derived node and s are in $P_n(s_n)$. Finally implement the routing algorithm proposed by Gates and Papadimitriu[2] for these two nodes.

The minimum length of the routing path is at most $d(P_{n-1}) + 2 = 5[(n - 1) + 1] / 3 + 2 = (5n / 3) + 2 = 5(n + 1) / 3 + (1 / 3) = d(P_n) + (1 / 3)$, and the time complexity is $O(n - 1) + 3 = O(n)$.

Theorem 3: Given an n-pancake graph P_n, there is

one faulty node, consider the routing algorithm from source node to destination node, the minimum length of the routing path is at most $d(P_n) + (4 / 3)$, and the time complexity is O(n).

Proof: Routing algorithm is one possibility of the above 11 cases, so the minimum length of the routing path is at most $(5n / 3) + 3 = 5(n + 1) / 3 + (4 / 3) = d(P_n) + (4 / 3)$, and the time complexity is O(n).

4 Conclusions

In this paper, we propose an efficient O(n) fault-tolerant node-to-node routing algorithm with one faulty node in n-pancake graph P_n. The minimum length of the routing path is at most $(5n / 3) + 3 = d(P_n) + (4 / 3)$. The related research comparisons are listed in Table 1.

References

[1] S.B. Akers and B. Krishnamurthy. A group-theoretic model for symmetric interconnection networks. *IEEE Trans. Computers, vol 38 (4), pp 555-566, 1989*.

[2] W.H. Gates and C.H. Papadimitriou. Bounds for sorting by prefix reversal. *Discrete Math. Vol 27, pp 47-57, 1979*.

[3] K. Kaneko, N. Sawada and S. Peng. Cluster fault-tolerant routing in pancake graphs. *Proc. the 19th IASTED conf. Parallel and Distributed Computing and Systems, pp 19-21, 2007*.

Table 1: The related research comparisons, one faulty node algorithms is proposed in this paper.

Faulty Node	No	One node	n – 2 nodes	n – 2 clusters
Time complexity	O(n)	O(n)	O(n)	O(n)
Length of routing path	$d(P_n)$	$d(P_n) + (4 / 3)$	$d(P_n) + (8 / 3)$	$d(P_n) + 7$

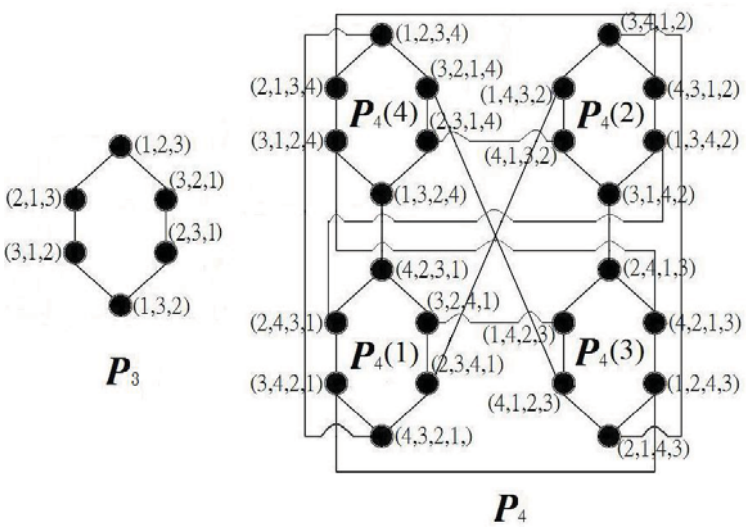

Figure 1: The example of pancake graphs P3 and P4, there are four sub-pancake graphs: P4(1), P4(2), P4(3), and P4(4) among P4.

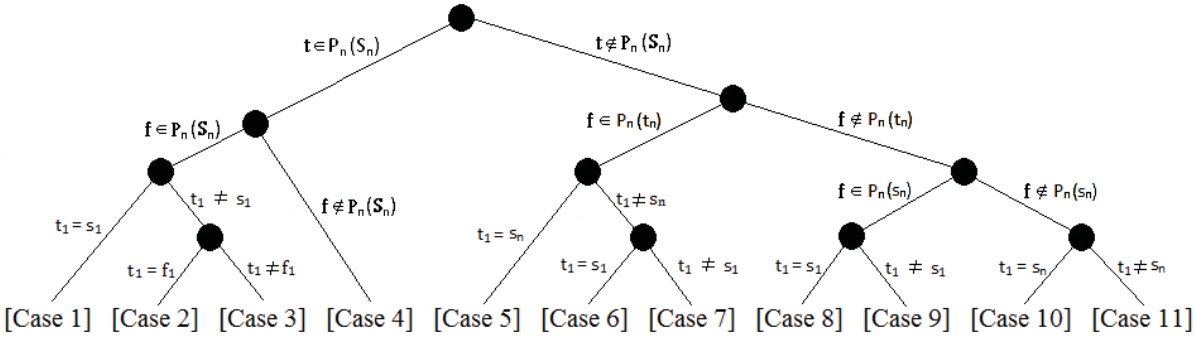

Figure 2: The total 11 cases of fault-tolerant routing algorithms in n-pancake graph with one faulty node. The source node s = (s1, s2, ⋯, sn), the destination node t = (t1, t2, ⋯, tn), and the faulty node f = (f1, f2, ⋯, fn).

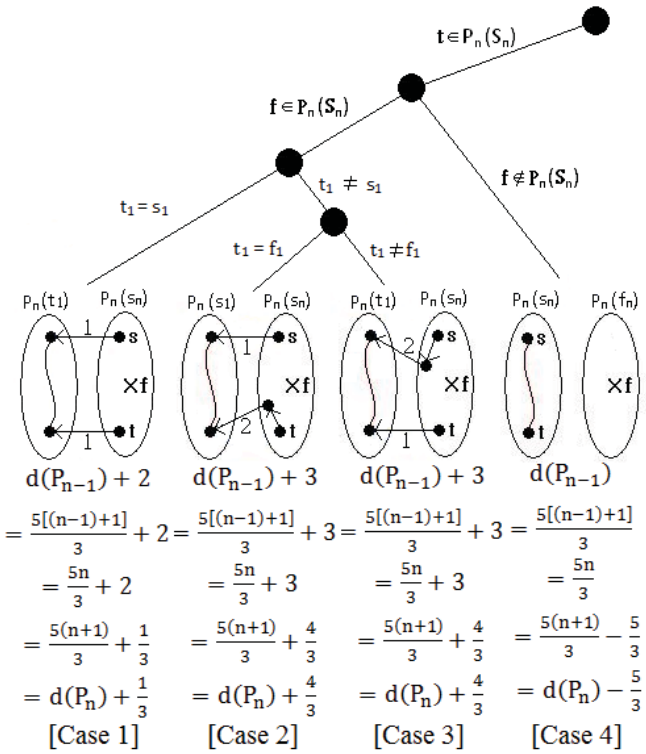

Figure 3: The Case 1-4 of the fault-tolerant routing algorithms from the source node s to the destination node t in n-pancake graph to avoid the faulty node f.

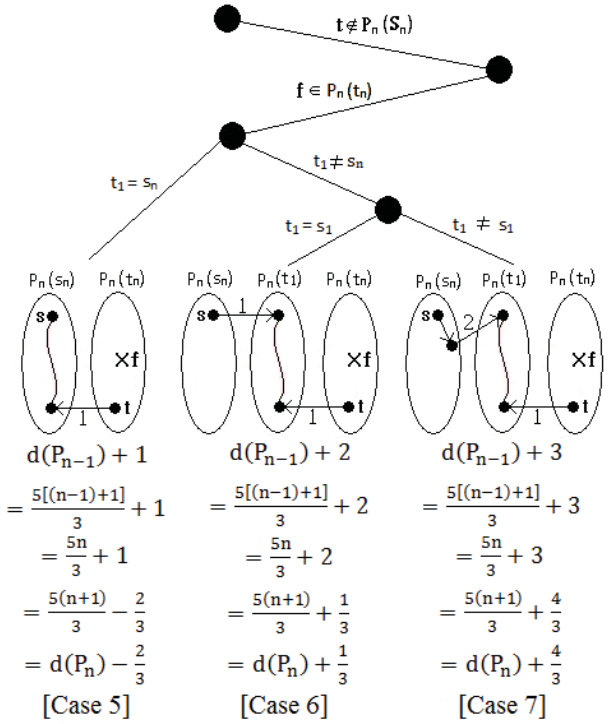

Figure 4: The Case 5-7 of the fault-tolerant routing algorithms from the source node s to the destination node t in n-pancake graph to avoid the faulty node f.

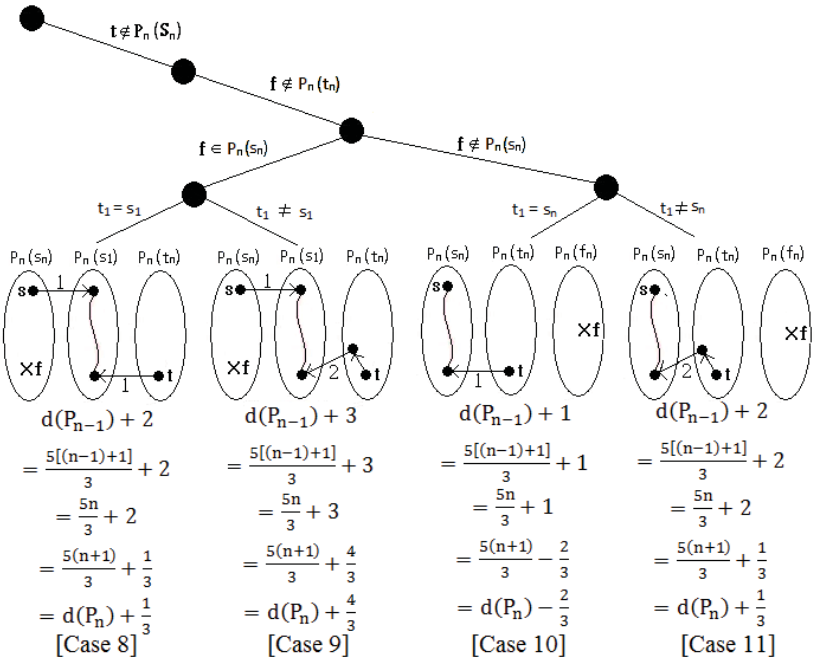

Figure 5: The Case 8-11 of the fault-tolerant routing algorithms from the source node s to the destination node t in n-pancake graph to avoid the faulty node f.