反転と転移によるゲノム再編成の近似アルゴリズム

松田 秀雄，　山中 啓之，　橋本 昭洋
{matsuda, h-yamana, hasimoto}@ics.es.osaka-u.ac.jp

〒560-8531 豊中市待兼山町 1-3
大阪大学 大学院基礎工学研究科 情報数理系専攻

近年，生物のゲノム上の遺伝子の並びを解析することにより，生物の進化過程を推定する手法が提案されている．この手法では，2 つの生物のゲノムで同一の遺伝子の並びが与えられたとき，ある生物の遺伝子並びをもう一方の生物の遺伝子並びに変換する最小の操作回数を求める．操作には任意個数の連続する遺伝子の転移およびそれらの順番の反転がある．この問題は NP-困難であると予想されているので，我々はこの問題に対する多項式時間近似アルゴリズムを開発し，このアルゴリズムの近似率が 2 であることを証明した．さらに，本アルゴリズムにより乱数で生成した遺伝子並びを変換したときの最適解との比較，および本アルゴリズムを最近その全 DNA 配列が決定されたいくつかの細菌のゲノムに対して適用した結果について報告する．

An Approximation Algorithm for Genome Rearrangements with Reversals and Transpositions

Hideo Matsuda,　Hiroyuki Yamanaka and Akihiro Hashimoto,
Department of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka, 560–8531

Recently a new approach has been proposed for inferring the evolutionary process of genomes based on comparison of gene orders. It searches for the minimum number of operations for sorting of a permutation of genes from one genome to another by reversals and transpositions. The complexity of the problem is conjectured as NP-hard. We developed a polynomial-time approximation algorithm for the problem and proved that the algorithm provides an approximation ratio of 2. We compare its performance with the optimum solution in randomly generated permutations, and show results by applying the algorithm to comparison between bacterial genomes whose complete DNA sequences have been determined recently.

1 Introduction

Recently, a new approach to the evolutionary analysis among species was proposed [1]. It focuses on comparison of not *gene sequences* but *gene orders* on whole DNA sequences (so called, *genome*) of different species, taking account of several genome-level mutations.

A combinatorial problem of *sorting by reversals* (corresponding to genome rearrangements by inversions) studied intensively. Kececioglu and Sankoff suggested the first performance guaranteed (2-approximation) algorithm for this problem [2]. Later Bafna and Pevzner improved the error bound to 7/4 [3]. The problem is shown to be NP-hard [4].

Bafna and Pevzner studied a similar *sorting by transpositions* problem. They gave a (3/2)-approximation algorithm [5]. Gu, Peng and Sudborough studied an extended problem that is *sorting by reversals and transpositions* simultaneously [6]. In the problem, they introduced three operations: reversal, transposition and *reversal+transposition* that inverts a segment of DNA and inserts it into another place on the same DNA sequence as one operation. They gave two approximation algorithms for this problem: a 2-approximation algorithm of which time complexity may not be bound by a polynomial of the length of the permutation, and a $2(1+1/k)$-approximation algorithm, where $k \geq 3$ is any fixed integer, which runs in polynomial time.

In this paper we treat a restricted version of the problem studied by Gu et al., which includes only reversal and transposition operations. A reversal+transposition operation is regarded as a combination of reversal and transposition operations. We will give a 2-approximation algorithm and performance results for sorting permutations generated randomly and gene orders of bacterial genomes that are completely determined recently.

2 Genome Rearrangements

In this problem, each gene is represented by an integer number. Since genes are oriented in a DNA sequence, each gene is described as a number with + or - sign. Thus the order of genes in a DNA sequence is represented by a *signed* permutation of the numbers.

Let $\Pi = (\pi_1 \pi_2 \cdots \pi_n)$ represent a signed permutation of integers 1 through n, where π_i is the number in the i-th position.

A *reversal* $r(i,j)$ of the interval $[i, j-1]$ is an inversion of the subsequence $\pi_i \pi_{i+1} \cdots \pi_{j-1}$ of Π $(1 \leq i < j \leq n+1)$ ($\Pi \cdot r(i,j) = (\pi_1 \cdots \pi_{i-1}\ -\pi_{j-1} \cdots -\pi_i\ \pi_j \cdots \pi_n)$).

A *transposition* $t(i,j,k)$ inserts an interval $[i, j-1]$ of Π between π_{k-1} and π_k $(1 \leq i < j \leq n+1, 1 \leq k \leq n+1)$ ($\Pi \cdot t(i,j,k) = (\pi_1 \cdots \pi_{i-1}\, \pi_j \cdots \pi_{k-1}\, \pi_i \cdots \pi_{j-1}\, \pi_k \cdots \pi_n)$

Bafna and Pevzner introduced the notion of *breakpoint graph* for representing the structure of the problem [3]. First, they introduced a breakpoint graph for *unsigned* permutation. Then they extended it for signed permutation. We follow the fashion.

Breakpoint graph for unsigned permutation: Add extra two numbers $\pi_0 = 0$ and $\pi_{n+1} = n+1$ into an unsigned permutation $\Pi = ((\pi_1 \pi_2 \cdots \pi_n)$ of $\{1, 2, ..., n\}$. Let $i \sim j$ if $|i-j| = 1$. A pair of consecutive elements π_i and π_{i+1} $(0 \leq i \leq n)$ of Π is called a *breakpoint* if $\pi_i \not\sim \pi_{i+1}$. A pair of non-consecutive elements π_i and π_j $(i \not\sim j)$ is called an *adjacency* if $\pi_i \sim \pi_j$. A breakpoint graph $G(\Pi) = (V, E)$ of Π is a graph such that each vertex $v \in V$ is a number which is in either a breakpoint pair or an adjacency pair and each edge $e \in E$ is a link between either a break point pair or an adjacency pair. Hereafter, we call a link between a breakpoint pair *b-edge* and a link between an adjacency pair *a-edge*. Every edge in the graph is involved in *alternating cycle*, namely every two consecutive edges are different types, either a-edge or b-edge (see Lemma 2). Hereafter we refer to alternating cycle as "cycle".

Breakpoint graph for signed permutation: For extending the notion of breakpoint graph for unsigned permutation to that for signed permutation, Bafna and Pevzner introduced a transformation to a signed permutation of size n to a corresponding unsigned permutation of size $2n$ [3]: a positive number $+i$ is replaced by two unsigned numbers, $2i-1$ and $2i$; whereas a negative number $-i$ is replaced by $2i$ and $2i-1$. After this transformation, breakpoint graph for signed permutation is constructed similarly to that for unsigned

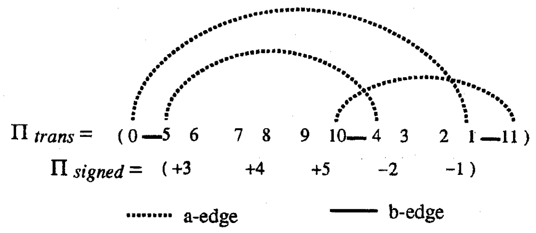
Π trans = (0—5 6 7 8 9 10—4 3 2 1—11)
Π signed = (+3 +4 +5 -2 -1)
a-edge
b-edge

Figure 1: A breakpoint graph of a signed permutation $\Pi = (+3\ +4\ +5\ -2\ -1)$.

permutation.

Figure 1 shows a breakpoint graph of a signed permutation $\Pi = (+3\ +4\ +5\ -2\ -1)$. In Fig. 1, its transformed unsigned permutation $\Pi_{trans} = (5\,6\,7\,8\,9\,10\,4\,3\,2\,1)$ and two extra numbers 0 and 11 are added at the both ends. Hereafter we refer to the transformed unsigned permutation (from a signed permutation) as "permutation".

3 Lower Bound

For a permutation Π, let $b(\Pi)$, $c(\Pi)$ and $d(\Pi)$ be the number of b-edges (i.e., breakpoints), the number of cycles and the distance between Π and I (i.e., the minimum number of operations that transforms Π into I), respectively. Denote $\Delta b = \Delta(\Pi, \rho) = b(\Pi) - b(\Pi\rho)$ (decrease in the number of breakpoints by an operation rho, either a reversal or a transposition), $\Delta c = \Delta c(\Pi, \rho) = c(\Pi) - c(\Pi\rho)$ (decrease in the number of cycles by rho).

Since a transposition may affect three b-edges, $-3 \leq \Delta b \leq 3$ by a transposition. Similarly, since a reversal may affect two b-edges, $-2 \leq \Delta b \leq 2$ by a reversal. We refer to these types of transpositions or reversals as *i-transposition* or *i-reversal* where $i = \Delta b$.

Thus trivial lower bound of the minimum number of operations for sorting a permutation Π is $b(\Pi)/3$. To conduct better lower bound, we introduce the following three lemmas on the property of breakpoint graph. The proofs of these lemmas are omitted due to space limitation in this paper.

Lemma 1 For every π_i $(0 \leq i \leq 2n+1)$ in $G(\Pi)$, there exist at most one b-edge and at most one a-edge incident with π_i.

Lemma 2 Every path along edges in $G(\Pi)$ forms an alternating cycle composed of the same number of a-edges and b-edges.

Lemma 3 Every cycle in $G(\Pi)$ has at least two a-edges and at least two b-edges.

Lemma 3 suggests the following theorem.

Theorem 1 $\Delta c \geq -2$.

Proof. The number of newly yielded b-edges (i.e., $-\Delta b$) is at most three in a transposition. Thus, by Lemma 3, only one new cycle can be created if every b-edge of the cycle is either of the new b-edges. Another possibility is the division of a cycle. A cycle can be divided into at most three cycles by using the three new b-edges. Thus the increase in the number of cycles (i.e., $-\Delta c$) is at most two (3 new cycles − 1 original cycle). □

From Theorem 1, the following theorem is derived.

Theorem 2 $\Delta b - \Delta c \leq 2$.

Proof. As mentioned above, $-3 \leq \Delta b \leq 3$. Thus we consider the following cases.

(i) $\Delta b = 3$

Only 3-transposition can remove three b-edges simultaneously. In this case, a cycle that has the three b-edges to be removed is eliminated. Namely, the number of cycles decreases by one (i.e., $\Delta c = 1$). Thus $\Delta b - \Delta c = 2$.

(ii) $\Delta b = 2$

Either 2-transposition or 2-reversal removes two b-edges. 2-transposition yields only one new b-edge. However at least two new b-edges are needed for constructing a new cycle or dividing a cycle into two cycles. Thus no cycles are not yielded by this operation, namely $\Delta c \geq 0$. Therefore $\Delta b - \Delta c \leq 2$. In the case of 2-reversal, a cycle that has the two b-edges to be removed is eliminated (i.e., $\Delta c = 1$). Thus $\Delta b - \Delta c = 1$.

(iii) $\Delta b = 1$

Either 1-transposition or 1-reversal removes one b-edge. 1-transposition yields two new b-edges. Thus a new cycle may be created or a cycle may be divided into two cycles by 1-transposition. This implies $\Delta c \geq -1$. Therefore $\Delta b - \Delta c \leq 2$. Similarly, 1-reversal yields one new b-edge. Thus no cycles are not yielded by this operation (i.e., $\Delta c \geq 0$). Therefore $\Delta b - \Delta c \leq 1$.

(iv) $\Delta b \leq 0$.
In this case, $\Delta b - \Delta c \leq 2$ is immediately derived from Theorem 1.

From (i) to (iv), this theorem is proved. □

Theorem 2 suggests the lower bound of this problem as follows.

Theorem 3 $d(\Pi) \geq (b(\Pi) - c(\Pi))/2$.

Proof. Let $t = d(\Pi)$. Denote $\rho_t, \rho_{t-1}, ..., \rho_1$ are operations that give the minimum number of operations. Then the transformation from Π into I is represented by a recurrence relation $\Pi_{(i-1)} = \Pi_{(i)}\rho_i$ $(1 \leq i \leq t)$ under $\Pi_{(t)} = \Pi$ and $\Pi_{(0)} = I$.

By Theorem 2,
$d(\Pi_{(i)}) = d(\Pi_{(i-1)}) + 1$
$\geq d(\Pi_{(i-1)}) + (\Delta b(\Pi_{(i)}, \rho_i) - \Delta c(\Pi_{(i)}, \rho_i))/2$
$= d(\Pi_{(i-1)})$
$+(b(\Pi_{(i)}) - b(\Pi_{(i-1)}) - (c(\Pi_{(i)}) - c(\Pi_{(i-1)})))/2$

This equation can be transformed into the following equation.
$d(\Pi_{(i)}) - (b(\Pi_{(i)}) - c(\Pi_{(i)}))/2$
$\geq d(\Pi_{(i-1)}) - (b(\Pi_{(i-1)}) - c(\Pi_{(i-1)}))/2$
$\geq \cdots \geq d(\Pi_{(0)}) - (b(\Pi_{(0)}) - c(\Pi_{(0)}))/2$

Since the breakpoint graph for the identity permutation I has no b-edges and no cycles, $d(\Pi_{(0)}) = b(\Pi_{(0)}) = c(\Pi_{(0)}) = 0$. Thus, $d(\Pi) \geq (b(\Pi) - c(\Pi))/2$ holds. □

4 Upper Bound

In Theorem 3, the equality holds when $\Delta b - \Delta c = 2$. To approach this condition, we have developed a greedy algorithm. For each step, the algorithm explores an operation that removes as many breakpoints as possible without decrease in the number of cycles. To do this, we introduce the notions of such operations.

Suppose a cycle $C = (V_C, E_C)$ in a breakpoint graph $G(\Pi)$. a reversal $r(i, j)$ is called *one-cycle reversal* if $(\pi_{i-1}, \pi_i) \in E_C$ and $(\pi_{j-1}, \pi_j) \in E_C$. Similarly, a transposition $t(i, j, k)$ is called *one-cycle transposition* if $(\pi_{i-1}, \pi_i) \in E_C$, $(\pi_{j-1}, \pi_j) \in E_C$ and $(\pi_{k-1}, \pi_k) \in E_C$. Note that all the edges are b-edges.

The following theorem gives the property of these operations.

Theorem 4 $\Delta c \leq 0$ for one-cycle 2-transposition, one-cycle 1-transposition and one-cycle.

Proof. Let ρ be any of these operations. Let C_1 and $C_i (2 \leq i \leq c(\Pi))$ be the cycle where the operation is performed and the other cycles in $G(\Pi)$, respectively. Since $C_1 \not\subseteq G(\Pi\rho)$ and $C_i \subseteq G(\Pi\rho)$, $c(\Pi\rho) \geq c(\Pi) - 1$. Since any of the operations yield at least one new b-edges, there exists at least one cycle except C_i in $G(\Pi\rho)$. This implies $c(\Pi\rho) \geq c(\Pi)$. Thus $\Delta c = c(\Pi) - c(\Pi\rho) \leq 0$. □

In the algorithm, the operation to be performed at a step is determined based on the following classification of the types of breakpoints.

Let (π_{i-1}, π_i) and (π_{j-1}, π_j) be consecutive two b-edges that are connected by an a-edge. There are four types depending on the position of the a-edge.

(a) (π_{i-1}, π_{j-1}) is an a-edge.
A reversal $r(i, j)$ is a one-cycle 1-reversal or 2-reversal.

(b) (π_{i-1}, π_j) is an a-edge.
If there exists another b-edge (π_{k-1}, π_k) $(j \leq k-1)$ in the same cycle, a transposition $t(j, k, i)$ is a one-cycle 1-transposition, 2-transposition or 3-transposition.

(c) (π_i, π_{j-1}) is an a-edge.
If there exists another b-edge (π_{k-1}, π_k) $(i + 1 \leq k \leq j - 1)$ in the same cycle, a transposition $t(k, j, i)$ is a one-cycle 1-transposition, 2-transposition or 3-transposition.

(d) (π_i, π_j) is an a-edge.
A reversal $r(i, j)$ is a one-cycle 1-reversal or 2-reversal.

In type (b) and (c), another b-edge is required in the same cycle. Thus it is troublesome when a cycle consists of only type (b) and (c) b-edges. Bafna and Pevzner call such a cycle *non-oriented cycle* [5]. A non-oriented cycle has one a-edge of type (b) (between the leftmost end and the rightmost end in permutation) and at least one a-edge of type (c). For example, a breakpoint graph of $\Pi = (0\,1\,2\,7\,8\,5\,6\,3\,4)$ has a non-oriented cycle composed of four edges: $(7, 6)$ (a-edge of type (c)), $(6, 3)$ (b-edge), $(3, 2)$ (a-edge of type (b)) and $(2, 7)$ (b-edge).

However, the next theorem dissolve this problem.

Theorem 5 Between any two consecutive b-edges in a non-oriented cycle, there exists at least one b-edge in another cycle.

Proof. Assume no b-edge exists between two consecutive b-edges (say, (π_{i-1}, π_i) and (π_{j-1}, π_j) $(i \leq j-1)$ in a non-oriented cycle. By definition of non-oriented cycle, the a-edge between these b-edges belongs to type (c). Thus $i \not\sim j-1$ and $\pi_i \sim \pi_{j-1}$ hold. By the assumption, a sub-permutation $(\pi_i \pi_{i+1} \cdots \pi_{j-1})$ is a sequence of ascending or descending order. Thus either $\pi_i \leq \pi_{j-1} - 2$ or $\pi_i \geq \pi_{j-1} + 2$. Thus $\pi_i \not\sim \pi_{j-1}$ but it contradicts that (π_i, π_{j-1}) is an a-edge. □

Thus, in type (c), there exists another b-edge (π_{k-1}, π_k) $(i+1 \leq k \leq j-1)$. Therefore at least 1-transposition can be carried out even in a non-oriented cycle.

The outline of our algorithm is as follows:

```
Algorithm 1
begin
  Construct G(Π);
  while there exist b-edges
  begin
    if one-cycle 3-transposition is executable
      do it;
    elseif one-cycle 2-transposition is executable
      do it;
    elseif one-cycle 2-reversal is executable
      do it;
    elseif one-cycle 1-transposition is executable
      do it;
    elseif one-cycle 1-reversal is executable
      do it;
    elseif a non-oriented cycle exists
      do 1-transposition for a non-oriented cycle
  end
end
```

In this algorithm, the number of iteration in while-loop is proportional to the number of b-edges and the process to check the condition in each if-statement needs time proportional to the number of b-edges. Since the number of b-edges is proportional to the size of permutation in the worst case, the time complexity of this algorithm is bound by $O(n^2)$ where n is the size of permutation.

The value of $\Delta b - \Delta c$ for each operation in this algorithm is: $\Delta b - \Delta c = 2$ in one-cycle 3-transposition, $\Delta b - \Delta c \geq 2$ in one-cycle 2-transposition by Theorem 4, $\Delta b - \Delta c = 1$ in one-cycle 2-reversal, $\Delta b - \Delta c \geq 1$ in one-cycle 1-transposition and one-cycle 1-reversal by Theorem 4 and $\Delta b - \Delta c \geq 1$ in 1-transposition in non-oriented cycle by Theorem 4.

Thus $\Delta b - \Delta c \geq 1$ holds for every operation in this algorithm. By similar reduction to the proof of Theorem 3, $d(\Pi) \leq b(\Pi) - c(\Pi)$ can be derived. Therefore we conclude that the approximation ration of this algorithm is 2.

5 Performance Results

For the performance evaluation of our algorithm, first we compared results with the optimum solution in random permutations. We generated 100 permutations of length 8. The optimum solution (i.e., the minimum number of operations transformed these permutations into the identity permutation) is computed by some exhaustive search method.

We also compared its performance with that of the first approximation algorithm by Gu et al. [6] (the 2-approximation algorithm of which time may not be bound by a polynomial of the length). Note that our implementation of Gu et al.'s algorithm does not handle reversal+transposition operation but only reversal and transposition operations.

Figure 2 shows the distribution of approximation ratios (the ratio of the number of operations computed by these algorithms to the optimum number). The 2-approximation algorithm developed by Gu et al. is referred as "Good Cycle", whereas our algorithm as "Proposed Method." The average approximation ratios are 1.4 in Gu et al.'s algorithm whereas 1.19 in our algorithm. In Fig. 2, all the solutions computed by both methods are converged within error bound of 2. Moreover our method achieves better performance than the algorithm by Gu et al.

For the comparison between real genomes, we used genomes as shown in Table 1. The genomes G1 and G2 belong to the same genus (one level higher layer than species); both G3 and G4 are methanogenic archaebacteria; and G5 and G6 are referred as closely related bacteria [7]. To extract identical pairs (more exactly, orthologous pairs) of genes, we performed pairwise global alignment for all combinations of gene sequences (in amino acid residues) based on the bi-directional best hit method [7]. Table 2 shows their results. The number of operations for transforming from one genome to another seems to fit biological knowledge. The execution times are measured on Sun SPARC-station 20 (SuperSPARC-II, clock 75 MHz).

6 Conclusions

We have developed a 2-approximation algorithms for genome rearrangements with reversals and transpositions. From comparison between our algorithm result and the optimum solution in random permutations, our algorithm achieves good performance whose average approximation ratio is 1.19. We also applied our algorithm to some bacterial genomes whose complete DNA sequences are determined recently. The results seem to fit biological knowledge, but its further evaluation remains as our future work.

References

[1] Sankoff, D. et al.: Gene Order Comparisons for Phylogenetic Inference: Evolution of the mitochondrial genome, *Proc. Natl. Acad. Sci. USA*, Vol. 89, pp. 6575–6579 (1992).

[2] Kececioglu, J. and Sankoff, D.: Exact and Approximation Algorithms for the Inversion Distance between Two Permutations, *Proc. 4th Ann. Symp. Combinatorial Pattern Matching*, LNCS No. 684, Springer Verlag, pp. 87–105 (1993).

[3] Bafna, V. and Pevzner P.: Genome Rearrangements and Sorting by Reversals, *SIAM J. Computing*, Vol. 25, No. 2, pp. 272–289 (1996).

[4] Caprara, A.: Sorting by Reversals is Difficult, *Proc. 1st Ann. Conf. Research in Computational Molecular Biology (RECOMB97)*, ACM Press, pp. 75–83 (1997).

[5] Bafna, V. and Pevzner P.: Sorting Permutations by Transpositions, *Proc. 6th ACM-SIAM Ann. Symp. on Discrete Algorithms*, pp. 614–623 (1995).

[6] Gu, Q.-P., Peng, S. and Sudborough, H.: Approximation Algorithms for Genome Rearrangements, *Proc. Genome Informatics 1996*, Universal Academy Press, pp. 13–22 (1996).

[7] Tatusov, R. L. et al.: Metabolism and evolution of *Haemophilus influenzae* deduced from a whole-genome comparison with *Escherichia coli*, *Current Biology*, Vol. 6, No. 3, pp. 279–291 (1996).

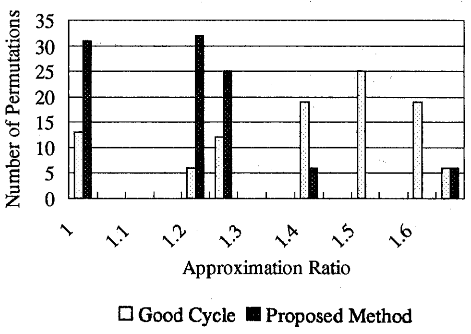

Figure 2: Distribution of the ratio of results to the optimum number

Table 1: Bacterial complete genomes used for the analysis of genome rearrangements

ID	Species	Number of genes
G1	*Mycoplasma genitalium*	468
G2	*Mycoplasma pneumoniae*	677
G3	*Methanococcus jannaschii*	1735
G4	*Methanobacterium thermoautotrophicum*	1871
G5	*Haemophilus influenzae*	1680
G6	*Escherichia coli*	4290

Table 2: Sorting results between bacterial genomes

Genomes	Number of matched genes	Number of operations	Exec. time
G1 vs G2	425	3	0.02
G3 vs G4	289	63	0.23
G5 vs G6	857	181	4.17

The execution time is measured in seconds.