Approximation Algorithms for Genome Rearrangements

(sorting signed permutations by reversals and transpositions)

Qian-Ping Gu [1] Shietung Peng [1] Hal Sudborough [2]
qian@u-aizu.ac.jp s-peng@u-aizu.ac.jp hal@utdallas.edu
[1] Department of Computer Software, The University of Aizu
Aizu-Wakamatsu, Fukushima, 965-80 Japan

[2] Department of Computer Science, The University of Texas at Dallas
Richardson, TX 75083 U.S.A.

Abstract

Recently, a new approach to analyze genomes evolving was proposed which is based on comparison of gene orders versus traditional comparison of DNA sequences (Sankoff et al, 1992). The approach is based on the global rearrangements (e.g., inversions and transpositions of fragments). Analysis of genomes evolving by inversions and transpositions leads to a combinatorial problem of sorting by reversals and transpositions, i.e., sorting of a permutation using reversals and transpositions of arbitrary fragments. The problem is conjectured as NP-hard. We study sorting of signed permutations by reversals and transpositions, a problem which adequately models genome rearrangements, as the genes in DNA are oriented. We establish a lower bound and give two algorithms for the problem. Based on the lower bound, we show that the first algorithm is a 2-approximation algorithm. The time complexity of the algorithm may not be bounded by Poly(n), where n is the length of the permutation to be sorted. Setting a time limit to the first algorithm, we get the second algorithm which is a $2(1+1/k)$-approximation one, where $k \geq 3$ is any fixed integer, and runs in Poly(n) time.

1 Introduction

Sequence comparison in computational molecular biology is a powerful tool for deriving evolutional and fundamental relationships between genes. However, classical alignment algorithms handle only local mutations (i.e., insertions, deletions, and substitutions of nucleotides) and ignore global rearrangements (i.e., inversions and transpositions of long fragments). Palmer and Herbon studied the rearrangements of mitochondrial genomes of *Brassica* (cabbage) and *Brassica campestris* (turnip) which are very closely related (many genes are $99\% - 99.9\%$ identical) [9]. They found that these molecules, which are almost identical in gene sequence, differ dramatically in gene order (see Figure 1). Other studies indicated that the classical methods of sequence comparison are not very useful to analyze highly rearranged genomes [6, 8]. Those works and many others showed that genome rearrangements is a common mode of molecular evolution in mitochondrial, chloroplast, viral and bacterial DNA [4, 2, 3, 1, 5].

Genomes evolve by inversions and transpositions as well as more simple operations of deletion, insertion and duplication of fragments. Analysis of genomes evolving involves solving a combinatorial "puzzle" to find a shortest series of reversals/transpositions from one genome into another. For genomes consisting of small number of "blocks" the shortest series may be found by the "pen-and-pencil" method. For example, Palmer et al, showed that Cabbage can be transformed into Turnip in three reversals as shown in Figure 1 [9, 5]. However, for genomes of large number of blocks, to find the solution is far beyond the possibilities of the "pen-and-pencil" methods. Recently, a computational approach to analyze the rearrangements of genomes was proposed by Sankoff et al, [10]. Representing the orders of genes by a permutation, analysis of genomes evolving leads to a combinatorial problem of sorting a permutation by reversals/transpositions.

Let $\pi = \pi_1\pi_2...\pi_n$ be a permutation of $\{1, 2, ..., n\}$. Sorting π by reversals/transpositions is to transform π into the identity $I = (12...n)$ by reversing and/or transposing arbitrary fragments of π. Assume that the orders of genes in two genomes are represented by π and I, respectively. The minimum number of operations (reversals/transpositions) of sorting π are used to measure the divergence between the genomes. However, it is not easy to find the minimum operations for sorting π and the problem is conjectured as NP-hard. Approximation algorithms for sorting of π have been studied extensively since 1992 [7, 3, 5, 2]. For a permutation π, let $d(\pi)$ be the minimum number of operations to sort π into I. An α-approximation algorithm for sorting π is an algorithm which finds a series of operations $\rho_1, ..., \rho_t$ such that $\rho_1, ..., \rho_t$ sort π into I and t satisfies $d(\pi) \leq t \leq \alpha d(\pi)$. Kececioglu and Sankoff gave a 2-approximation algorithm for sorting of π by reversals only [7]. The error bound of 2 was improved to (7/4) by Bafna and Penvzer [2]. A (3/2)-approximation algorithm for sorting of π by transposition only was given by Bafna and Penvzer [3].

A *signed permutation* is a permutation π on $\{1, 2, ..., n\}$ with $+$ or $-$ sign associated with every element π_i of π. For example, $(+1-5+4-3+2)$ is a signed permutation of $\{1, 2, 3, 4, 5\}$. The identity of signed permutations is $(+1+2+...+n)$. Signed permutations are more relevant to genomes rearrangements, since genes are oriented in DNA sequences. Hannenhalli and Penvzer gave an algorithm which finds the minimum number of reversals for a signed permutation [5].

Bafna and Penvzer suggested the sorting by reversals and transformations simultaneously as an approach for understanding the genomes rearrangements related to mammalian genome evolution, viral evolution, and so on [3]. We consider sorting of signed permutations by reversals and transpositions simultaneously. For a permutation $\pi = \pi_1\pi_2...\pi_n$, (π_i, π_{i+1}) is called a *breakpoint* if $|\pi_i - \pi_{i+1}| \neq 1$. Obviously, any reversal/transposition can reduce at most 3 breakpoints. Let $b(\pi)$ be the number of breakpoints in π. Then $b(\pi)/3$ is a trivial lower bound for sorting π into I (I has no breakpoint). It was also known that some permutations of n elements take at least $n/2$ operations to be sorted [11]. In this paper, we establish a non-trivial generalized lower bound on the number of operations for sorting of signed permutations by reversals and transpositions simultaneously. Then we give two sorting algorithms. Based on the established lower bound, we show that the first algorithm is a 2-approximation algorithm. The time complexity of the algorithm may not be bounded by Poly(n), where n is the length of the permutation to be sorted. Setting a time limit to the first algorithm, we get the second algorithm which runs in Poly(n) time and is a $2(1+1/k)$-approximation algorithm, where $k \geq 3$ is any fixed integer. Some other works related to the above are: Sudborough gave an algorithm for sorting of an unsigned permutation of by reversals and transpositions [11]. The algorithm

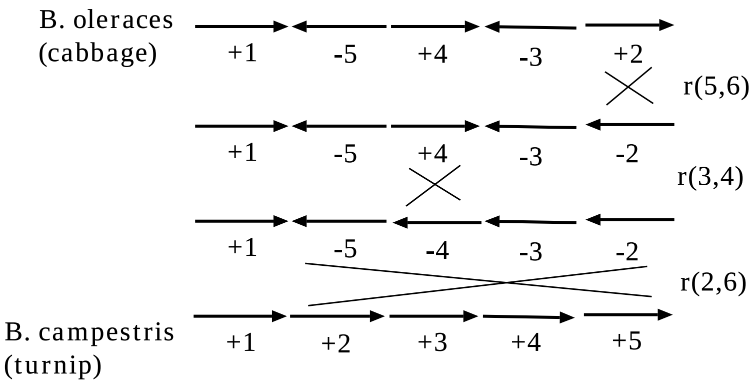

Figure 1: "Transformation" of cabbage into turnip.

sorts a permutation of length n in at most $2n/3$ operations.

The rest of this paper is organized as follows. In the next section, we give the definitions and notations of the paper. Section 3 gives the lower bound on the number of operations for sorting of signed permutations. We show the approximation algorithms in Section 4. The final section concludes the paper.

2 Preliminaries

Let $\pi = (\pi_1\pi_2...\pi_n)$ be a permutation of $\{1, 2, ..., n\}$. For $1 \leq i < j \leq n+1$, a *reversal* $r(i, j)$ is the permutation

$$(1, 2, ..., i-1, \mathbf{j-1}, ..., \mathbf{i+1}, \mathbf{i}, j, ..., n).$$

$\pi \cdot r(i, j) = (\pi_1...\pi_{i-1}\pi_{j-1}...\pi_{i+1}\pi_i\pi_j...\pi_n)$, i.e., $\pi \cdot r(i, j)$ has the effect of reversing the order of $\pi_i, \pi_{i+1}, ..., \pi_{j-1}$. For $1 \leq i < j \leq n+1$ and $1 \leq k \leq n+1$ with $k \notin [i, j]$, a *transposition* $t(i, j, k)$ is the permutation

$$(1, ..., i-1, \mathbf{j}, ..., \mathbf{k-1}, \mathbf{i}, ..., \mathbf{j-1}, k, ..., n).$$

$\pi \cdot t(i, j, k) = \pi_1...\pi_{i-1}\pi_j...\pi_{k-1}\pi_i...\pi_{j-1}\pi_k...\pi_n$, i.e., $\pi \cdot t(i, j, k)$ has the effect of moving $\pi_i\pi_{i+1}...\pi_{j-1}$ to a new location of π between π_{k-1} and π_k. For $1 \leq i < j \leq n+1$ and $1 \leq k \leq n+1$ with $k \notin [i, j]$, a *reversal+transposition* $rt(i, j, k)$ is the permutation

$$(1, ..., i-1, \mathbf{j}, ..., \mathbf{k-1}, \mathbf{j-1}, ..., \mathbf{i}, k, ..., n).$$

$\pi \cdot rt(i, j, t) = \pi_1...\pi_{i-1}\pi_j...\pi_{k-1}\pi_{j-1}...\pi_{i+1}\pi_i\pi_k...\pi_n$, i.e., $\pi \cdot rt(i, j, k)$ has the effect of reversing $\pi_i\pi_{i+1}...\pi_{j-1}$ and then moving $\pi_{j-1}...\pi_i$ to a new location of π between π_{k-1} and π_k. We will call the reversal, transposition, and reversal+transposition *operations*.

Example 1: Let $\pi = (14352)$. Then $\pi \cdot r(1, 4) = (34152)$, $\pi \cdot t(1, 4, 5) = (51432)$, and $\pi \cdot rt(1, 4, 5) = (53412)$.

The distance between two permutations π and σ is the minimum number of operations $\rho_1, ..., \rho_t$ such that $\pi \cdot \rho_1 \cdot \rho_2 \cdots \rho_t = \sigma$. Note that the distance between π and σ equals to that

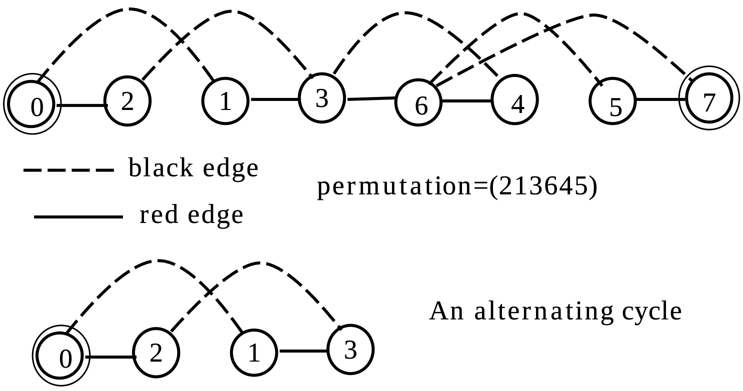

Figure 2: The breakpoint graph $G(\pi)$ of $\pi = (213645)$.

between $\sigma^{-1}\pi$ and the identity $I = (12...n)$. Thus, we only concentrate on finding the distance $d(\pi)$ between π and I.

A *signed permutation* is a permutation π on $\{1, 2, ..., n\}$ with $+$ or $-$ sign associated with every element π_i of π. For example, $(+1 - 5 + 4 - 3 + 2)$ is a signed permutation. The identity $I = (+1 + 2... + n)$. A reversal $r(i, j)$ on a signed permutation changes both the order and the signs of the elements within the fragment $\pi_i \pi_{i+1} ... \pi_{j-1}$ (see Figure 1). In this paper, we are interested in finding the minimum number of operations to sort a signed permutation into the identity $(+1 + 2... + n)$.

Bafna and Penvzer introduced the notion of *breakpoint graph* in their study for sorting by reversals only [2]. Our argument is also based on *breakpoint graph*, though we look at a different property of the graph.

Let π be an arbitrary unsigned permutation. Extend $\pi = \pi_1 \pi_2 ... \pi_n$ by adding $\pi_0 = 0$ and $\pi_{n+1} = n + 1$. Let $i \sim j$ if $|i - j| = 1$. We call a pair of consecutive elements π_i and π_{i+1} an *adjacency* if $\pi_i \sim \pi_{i+1}$, otherwise a *breakpoint*. Define a *breakpoint graph* $G(\pi)$ of π as follows [2]: There are $n + 2$ nodes $0, 1, 2, ..., n, n + 1$ in $G(\pi)$. There is a black edge between i and j if $i \sim j$ and i, j are not consecutive in π. There is a red edge between i and j if (i, j) is a breakpoint. The graph $G(\pi)$ for $\pi = (213645)$ is given in Figure 2. Notice that number of black edges equals to the number of red edges in $G(\pi)$, and equals to the number of breakpoints in π. The *breakpoint* graph $G(I)$ for the identity I has no egde.

A sequence of distinct nodes $v_1, v_2, ..., v_m$ is called a *segment* in a graph G if $(v_i, v_{i+1}) \in E(G)$ for $1 \leq i \leq m - 1$. A sequence of nodes $v_1, v_2, ..., v_m = v_1$ is called a *cycle* in a graph G if $(v_i, v_{i+1}) \in E(G)$ for $1 \leq i \leq m - 1$. A cycle/segment in a breakpoint graph G is called *alternating* if the colors of every two consecutive edges of this cycle are distinct. For example, cycle $(0, 2)(2, 3)(3, 1)(1, 0)$ of the graph $G(\pi)$ in Figure 2 is alternating.

Define a transformation from a signed permutation π of n elements to an unsigned permutation π' of $2n$ elements as follows [2]: replace $+i$ with $(2i - 1, 2i)$ and replace $-i$ with $(2i, 2i - 1)$ for $1 \leq i \leq n$. Notice that the identity $I = (+1 + 2... + n)$ is transformed into the unsigned identity $I' = (1234...(2n - 1)2n)$. Given any sequence of operations $\rho_1, ..., \rho_t$ which transforms π into σ, obviously, there is a sequence $\rho'_1, ..., \rho'_t$ which transforms π' into σ'. On the other hand, for any sequence of operations $\rho'_1, ..., \rho'_t$ transforming π' into σ' such that no

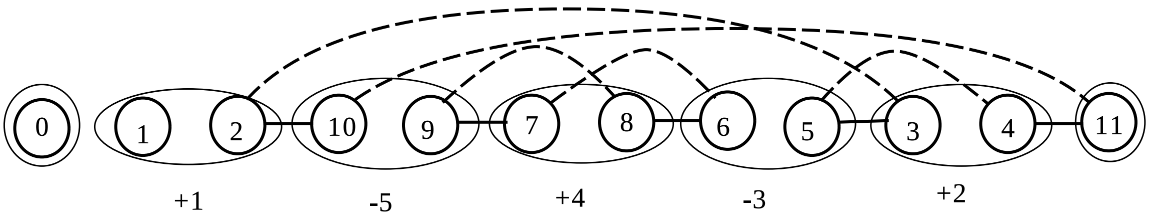

Figure 3: The breakpoint graph $G(\pi)$ of $\pi = (+1 - 5 + 4 - 3 + 2)$.

operation breaks any pair of $(2i-1, 2i)$ or $(2i, 2i-1)$ for $1 \leq i \leq n$, then there is a sequence of operations $\rho_1, ..., \rho_t$ that transforms π into σ. In what follows, we assume that any operation on the transformed unsigned permutation never breaks any pair of $(2i-1, 2i)$ or $(2i, 2i-1)$. Based on this assumption, the signed permutation π and the transformed unsigned permutation π' are equivalent for our purpose. When we refer to the breakpoint graph of a signed permutation, it is implied that we refer to the breakpoint graph of the transformed unsigned permutation. Figure 3 gives the breakpoint graph of $G(\pi)$ for $\pi = (+1 - 5 + 4 - 3 + 2)$.

3 Lower Bound

We first show some important properties of the breakpoint graph of a signed permutation.

Lemma 1 *For the breakpoint graph $G(\pi)$ of a signed permutation π, (1) the black degree and the red degree of each node in $G(\pi)$ are the same and equal to either 0 or 1; (2) each connected component of $G(\pi)$ is an alternating cycle; and (3) each alternating cycle has at least 2 black (red) edges.*

Proof: The lemma holds immediately from the definition of $G(\pi)$. $\square$

Let π be a permutation, ρ an operation, and $\pi' = \pi \cdot \rho$. Let $b(\pi)$ and $b(\pi')$ be the breakpoints in π and π', respectively. Then we have $|b(\pi) - b(\pi')| \leq 3$. From this, a trivial lower bound on sorting any permutation is $b(\pi)/3$. Now, we give a better lower bound on sorting signed permutations. The new lower bound is motivated by the following observation. Given a π, one operation ρ can reduce at most 3 breakpoints and if ρ reduces 3 breakpoints, it must be the case as shown in Figure 4 (b and c in the top row of the figure can be exchanged). In this case, the related elements form a cycle with three red edges. In the other cases, one operation can reduce at most 2 breakpoints.

Call an alternating cycle a *k-cycle* if it has k red edges. Call a k-cycle a good cycle, if there are at most $\lfloor (k-1)/2 \rfloor$ operations that remove the cycle from G. Intuitively, a good cycle is a cycle with $2j+1$ red edges ($j \geq 1$) that can be removed by j operations. Let $c(\pi)$ be the number of good cycles in $G(\pi)$. Our goal is to prove that $d(\pi) \geq (b(\pi) - c(\pi))/2$. The following theorem is the key to get the lower bound.

Theorem 2 *For a signed permutation π and an operation ρ with $\pi' = \pi \cdot \rho$, let $G = G(\pi)$ and $G' = G(\pi')$, b and b' be the number of red edges in G and G', and c and c' be the number of good cycles in G and G', respectively. Then $(b - b') + (c' - c) \leq 2$.*

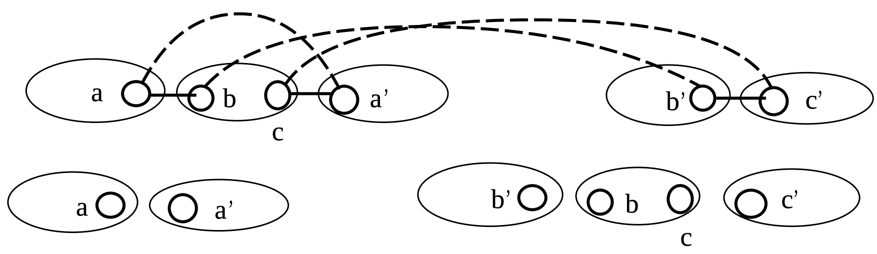

Figure 4: Removing three breakpoints by one operation.

Proof: As shown in Figure 5, we consider an operation ρ as a process that removes some red (black) edges from G and then add some red (black) edges into G to transform G into G'. We say an edge (i, j) is removed if $(i, j) \in G$ and $(i, j) \notin G'$. We say an edge is added if $(i, j) \notin G$ and $(i, j) \in G'$. Notice that removing edges breaks cycles into segments and adding edges joins segments into cycles. If a black edge (i, j) is removed by ρ from a cycle, then the adjacent red edges (k, i) and (j, l) must be removed by ρ as well. Therefore, a segment reduced by ρ must have two black edges at the ends (see Figure 5). From this, we conclude that

(a) to join one segment into one cycle, we need adding at least one red edge and

(b) to joint two segments into one cycle, we need adding at least two red edges.

Let $\mathbf{D}$ and $\mathbf{D}'$ be the sets of cycles in G and G', respectively. Call a cycle C a *new cycle*, if $C \in \mathbf{D}'$ and $C \notin \mathbf{D}$. Obviously, a new cycle has at least one added red edge. By Lemma 1, an operation ρ breaks some cycles in G into segments by removing certain edges first, and then joins every segments into *new cycles*. Also, operation ρ never adds an edge to a cycle which is not broken. In what follows, we only concentrate on the changes of red edges. Obviously, an operation ρ removes at most 3 red edges and adds at most 3 red edges, and $|b - b'| \leq 3$. When we say remove/add red edges, we mean the red (black) edges are removed/added by an operation ρ. The theorem is proved on all the values of $b - b'$ case by case.

Case $b - b' = 3$:

In this case, ρ removes three red edges from G. Since no red edge is added, we can not leave any alternating segment after the removing due to Lemma 1. Assume the three removed red edges are not in the same cycle. Then there is a cycle which has one removed red edge. From (3) of Lemma 1, removing the red edge from the cycle will produce an alternating segment, a contradiction to Lemma 1. So, the three removed cycles are in the same cycle C. Similarly, C must be a 3-cycle. Since C is eliminated by ρ, C is a good cycle and $(b - b') + (c' - c) \leq 2$.

Case $b - b' = 2$:

In this case, the number of new cycle is at most 1. If there is no new cycle, then $c' \leq c$ and $(b - b') + (c' - c) \leq 2$.

So we assume that there is one new cycle C'. $b - b' = 2$ implies that at most one red edge is added. Therefore, from (a) and (b), C' is obtained by joining the only segment P which is reduced by removing red edges from a cycle C of G. From Lemma 1 and $b - b' = 2$, it is easy to see that if C' is a k-cycle, then C is a $(k+2)$-cycle. If C' is a good cycle, then there are $\lfloor (k-1)/2 \rfloor$ operations that removes C' which implies there are $\lfloor (k+1)/2 \rfloor$ operations that

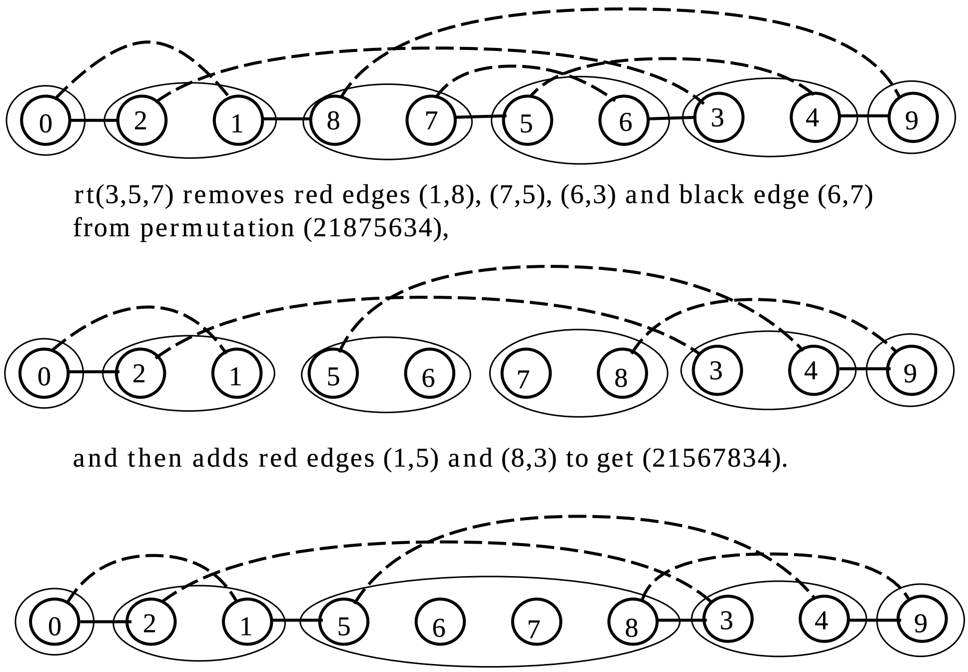

Figure 5: Removing/adding edges from/to breakpoint graph.

removes C, i.e., C is good cycle as well. Thus, $c' \leq c$ and $(b-b')+(c'-c) \leq 2$.

Case $b - b' = 1$:

In this case, the number of new cycles is at most 2. If there is at most one new cycle, then $c' \leq c+1$ and $(b-b')+(c'-c) \leq 2$.

Assume there are two new cycles. Let C_1 and C_2 be the new cycles. From $b - b' = 1$, two red edges are added and each new cycle has one added red edge. From (a) and (b), C_1 and C_2 are obtained by joining segments P_1 and P_2, respectively. Assume P_1 and P_2 are reduced from different cycles, then at least one segment, say P_1, is reduced from a cycle C by removing one red edge. This implies that when we join P_1 into a cycle C_1, $C_1 = C$, contradicting with C_1 a new cycle. Therefore, P_1 and P_2 are reduced from the same cycle C by removing some red edges. By a similar argument as that in Case $b - b' = 2$, if C_1 and C_2 are good cycles then C is a good cycle as well, which implies $c' \leq c+1$ and $(b-b')+(c'-c) \leq 2$.

Case $b - b' = 0$:

In this case, there are at most three new cycles. If there are at most two new cycles, then $c' - c \leq 2$ and $(b-b')+(c'-c) \leq 2$.

So, we assume there are three new cycles. Let C_1, C_2, and C_3 be the three new cycles obtained by joining segments P_1, P_2, and P_3, respectively. By a similar argument as that of Case $b - b' = 1$, P_1, P_2, and P_3 are reduced from the same cycle C by removing some red edges (see Figure 6). And if C_1, C_2, and C_3 are good cycles, then C is a good cycle. Therefore, $c' \leq c+2$ and $(b-b')+(c'-c) \leq 2$.

Case $b - b' \leq -1$:

Since there are at most three new cycles, $c' \leq c+3$ and $(b-b')+(c'-c) \leq 2$. □

From Theorem 2, we can get our lower bound.

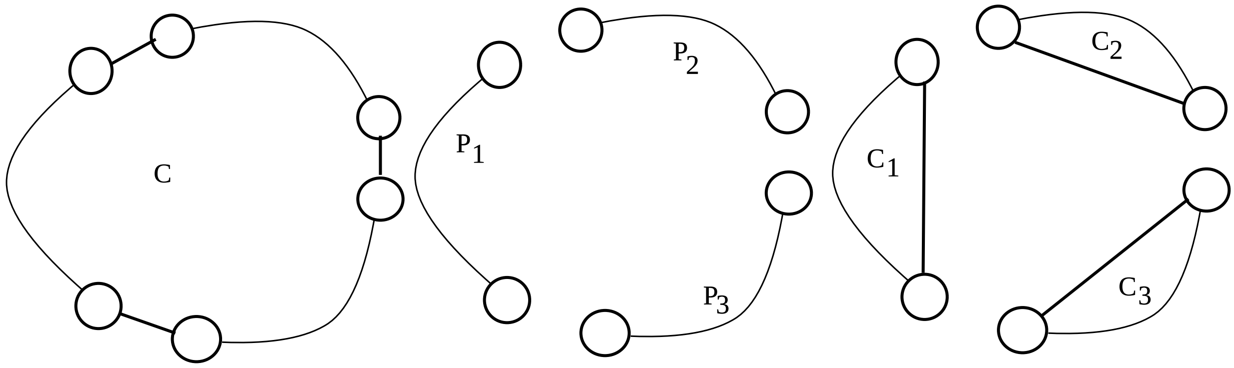

Figure 6: Breaking one cycle into three.

Theorem 3 *For a signed permutation π, $d(\pi) \geq (b(\pi) - c(\pi))/2$.*

Proof: Let $\rho_1, ..., \rho_t$ be a shortest series of operations transforming π into the identity permutation I. Denote $\pi_{i-1} = \pi_i \cdot \rho_i$ for $1 \leq i \leq t$ ($\pi_0 = I$) and apply Theorem 2 for π_i and ρ_i, we have

$$\begin{aligned} d(\pi_i) &= d(\pi_{i-1}) + 1 \\ &\geq d(\pi_{i-1}) + (b(\pi_i) - b(\pi_{i-1}) + c(\pi_{i-1}) - c(\pi_i))/2. \end{aligned}$$

From this and $d(\pi_0) = b(\pi_0) = c(\pi_0) = 0$, we get

$$\begin{aligned} d(\pi_i) - (b(\pi_i) - c(\pi_i))/2 &\geq d(\pi_{i-1}) - (b(\pi_{i-1}) - c(\pi_{i-1}))/2 \\ &\geq ... \geq d(\pi_0) - (b(\pi_0) - c(\pi_0))/2 = 0. \end{aligned}$$

Substituting $i = t$, the theorem holds. □

Since a good cycle has at least three breakpoints, $c(\pi) \leq b(\pi)/3$. Therefore, $(b(\pi) - c(\pi))/2 \geq b(\pi)/3$ for any signed permutation π. On the other hand, based our lower bound, it is easy to find permutations of n elements which take at least $n/2$ operations to be sorted by checking the breakpoint graphs. Thus, $(b(\pi) - c(\pi))/2$ gives a better measure for the lower bound on $d(\pi)$.

4 The Algorithms

We first give an algorithm which sorts a signed permutation π into the identity I by at most $b(\pi) - 2c(\pi)$ operations. From the lower bound $(b(\pi) - c(\pi))/2$ of the last section, we conclude the algorithm is a 2-approximation algorithm. The outline of the algorithm is given in Figure 7.

Let C be a good cycle with $2j + 3$ breakpoints ($j \geq 0$). Then, we can remove $2j$ breakpoints from C in j operations and the rest 3 breakpoints in one operation. For the breakpoints not in a good cycle, we remove at least one breakpoint by one operation. From these, algorithm SORT1 transforms π to I in at most $c(\pi) + (b(\pi) - 3c(\pi)) = b(\pi) - 2c(\pi)$ operations, which implies

Algorithm SORT1(π);
begin
  Construct $G(\pi)$ and let $C_1, ..., C_r$ be the alternating cycles of $G(\pi)$;
  For $1 \leq i \leq r$ and C_i a good cycle
    remove $2j+1$ breakpoints in C_i by j operations;
  Remove the other breakpoints;
end.

Figure 7: Sorting a signed permutation.

Algorithm SORT2(π, k);
begin
  Construct $G(\pi)$ and let $C_1, ..., C_r$ be the alternating cycles of $G(\pi)$;
  For $1 \leq i \leq r$
    if C_i has at most k breakpoints and C_i a good cycle
      remove $2j+1$ breakpoints in C_i by j operations;
  Remove the other breakpoints;
end.

Figure 8: Sorting a signed permutation in poly(n) time.

$d(\pi) \leq b(\pi) - 2c(\pi)$. From $d(\pi) \geq (b(\pi) - c(\pi))/2$ and $\frac{b(\pi)-2c(\pi)}{(b(\pi)-c(\pi))/2} \leq 2$, $b(\pi) - 2c(\pi) \leq 2d(\pi)$, i.e., algorithm SORT1 is a 2-approximation algorithm.

Given a k-cycle C, we do not have a poly(k) time algorithm to check if C is a good cycle (a brute-force algorithm works but takes exponential time). Therefore, the run time of algorithm SORT1 may not be bounded by Poly(n) if there is a long cycle in $G(\pi)$.

Now, we revise algorithm SORT1 a bit to get a more time efficient algorithm SORT2 by sacrificing slightly the guaranteed error bound. Let $k \geq 3$ be a fixed integer, algorithm SORT2 checks cycles with at most k red edges to find good cycles. The algorithm is given in Figure 8.

Let $c_k(\pi)$ be the number of good cycles with at most k breakpoints in $G(\pi)$. Then algorithm SORT2 transforms π into I in at most $b(\pi) - 2c_k(\pi)$ steps, i.e., $d(\pi) \leq b(\pi) - 2c_k(\pi)$. Since $c(\pi) - c_k(\pi)$ is the number of good cycles each of which has at least $k+1$ breakpoints, $c(\pi) - c_k(\pi) \leq b(\pi)/(k+1)$. Following a detailed calculation, $b(\pi) - 2c_k(\pi) \leq 2(1 + 1/k)d(\pi)$. That is, algorithm SORT2 is a $2(1+1/k)$-approximation algorithm. Obviously, algorithm SORT2 runs in Poly(n) time for any constant k.

5 Conclusional Remarks

Computational approaches provide efficient tools for large-scale comparative genetic mapping which offers exciting prospects for understanding genomes evolution. This paper gives the first steps for computing the distance between genomes in the sense of reversals/transpositions rearrangements. We expect our algorithms calculates the reversal/transposition distance between genomes much more accurately for the practical data and could be used for the comparasion between genomes of large size that is beyond the possibility of the pen-and-pencil method. How to check a cycle a good cycle seems the bottleneck of the algorithms of this paper. Future works include developing heuristic methods for finding good cycles for practical data, and reducing the error bound (currently 2 or $2(1 + 1/k)$) further.

References

[1] V. Bafna and P. Pevzner. Sorting by reversals: genome rearrangements in plant organelles and evolutionary history of x chromosome. *Mol. Biol. and Evol.*, 12:239–246, 1995.

[2] V. Bafna and P. Pevzner. Genome rearrangements and sorting by reversals. *SIAM J. on Computing*, 25(2):272–289, 1996.

[3] V. Banfa and P. Pevzner. Sorting permutations by transpositions. In *Proc. of 6th ACM-SIAM Annual Symposium on Discrete Algorithms*, pages 614–623, 1995.

[4] N. Franklin. Conservation of genome form but not sequence in the transcription antitermination determinants of bacteriophages λ, $\phi 21$, and p22. *Jour. of Molecular Evolution*, 181:75–94, 1985.

[5] S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutation by reversals). In *Proc. of 27th ACM Symposium on Theory of Computing (STOC'95)*, pages 178–189, 1995.

[6] S. Karlin, E.S. Mocarski, and G.A. Schachtel. Molecular evolution of herpesviruses: genomic and protein sequence comparisons. *Jour. of Virology*, 68:1886–1902, 1994.

[7] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for the inversion distance between two permutations. In *Proc. of the 4th Annula Symposium on Combinatorial Pattern Matching, Lecture Notes in Computer Science 684*, pages 87–105 (Extended version has appeared in Algorithmica 13:180–210, 1995), 1993.

[8] D.J. McGeoch. Molecular evolution of large dna viruses of eukaryotes. *Seminars in Virology*, 3:399–408, 1992.

[9] J.D. Palmer and L.A. Herbon. Plant mitochondrial dna evolves rapidly in structure, but slowly in sequence. *Jour. of Molecular Evolution*, 27:87–97, 1988.

[10] D. Sankoff, G. Leduc, N. Antoine, B. Paquin, B.F. Lang, and R. Cedergren. Gene order comparasions for phylogenetic inference: evolution of the mitochondrial genome. In *Proc. of Natl. Acad. Sci. USA,89*, pages 6575–6579, 1992.

[11] H. Sudborough. Permutations, pancakes, and phylogeny. Manuscript.