國 立 交 通 大 學

生物資訊及系統生物研究所

碩 士 論 文

利用剪下-圓形化-線性化-插入的運算排序基因體

Sorting Genomes Using Cut-Circularize-Linearize-and-Paste Operations

研 究 生：黃亙亘

指導教授：邱顯泰 博士

盧錦隆 博士

中華民國 一 百 年 七 月

利用剪下-圓形化-線性化-插入的運算排序基因體

Sorting Genomes Using Cut-Circularize-Linearize-and-Paste Operations

研 究 生：黃互亘　　Student：Keng-Hsuan Huang

指導教授： 邱顯泰 博士　Advisor：Dr. Hsien-Tai Chiu

盧錦隆 博士　　Dr. Chin Lung Lu

國 立 交 通 大 學

生 物 資 訊 及 系 統 生 物 研 究 所

碩 士 論 文

A Thesis Submitted to Institute of Bioinformatics

College of Biological Science and Technology

National Chiao Tung University in partial Fulfillment of the Requirements for the Degree of Master in

Biological Science and Technology

July 2011, Hsinchu, Taiwan

中文摘要

在演化的過程中，基因體可能會經歷一些被稱為基因體重組的大規模突變。隨著愈來愈多的基因體被完整地定序，基於基因次序分析的基因體重組研究在演化樹的重建上扮演著重要的角色。在過去，有許多傳統的重組運算已被提出以評估兩個相關基因體基因次序的演化距離，例如反轉、移位、區塊互換、易位、分裂與融合。通常基因體重組的計算研究被定義成一個利用重組運算來排序一個排列的問題。在這個論文中，我們介紹一個利用剪下-圓形化-線性化-插入 (簡稱 CCLP)的運算來進行排序的問題，目的是要找出最少次數的 CCLP 運算來排序一個表示一條染色體的有號整數排列。CCLP 運算是一種基因體重組的運算，它會把染色體的某一個片段剪下，然後再把剪下的片段圓形化成一個暫時性的環狀染色體，接著再把這個環狀染色體線性化為一條線性染色體，最後再把線性化的染色體貼回到原來的染色體中，但在線性染色體被貼回去之前允許它被反轉。CCLP 運算可以模擬一些上述為人所熟知的重組，例如反轉、移位與區塊互換，以及其他未在生物文獻中被報導的重組。為了與反轉作區別，我們把其它的 CCLP 運算稱為非反轉的 CCLP 運算。最後，當反轉與非反轉 CCLP 運算的權重比為 1:2 時，我們提出一個時間複雜度為 $O(\delta n)$ 的演算法來解決有加權重的CCLP運算排序問題，其中 n 為一條染色體內的基因個數，而 δ 為排序過程中所需的 CCLP 運算次數。

Abstract

During evolution, genomes may undergo large-scale mutations called as genome rearrangements. With more and more genomes have been sequenced completely, genome rearrangement studies based on genome-wide analysis of gene orders play an important role in reconstructing phylogenetic trees. In the past, a variety of traditional rearrangement operations, such as reversals, transpositions, block-interchanges, translocations, fissions and fusions, have been proposed to evaluate the evolution distance between two related genomes in gene order. Usually, the computational studies of genome rearrangements are formulated as a problem of sorting a permutation by rearrangement operations. In this thesis, we introduce a problem, called as a sorting problem by cut-circularize-linearize-and-paste (CCLP) operations, which aims to find a minimum number of CCLP operations to sort a signed permutation representing a chromosome. The CCLP operation is a genome rearrangement operation that cuts a segment out of a chromosome, circularizes the segment into a temporary circle, linearizes the temporary circle as a linear segment, and possibly inverts the linearized segment and pastes it into the remaining chromosome. The CCLP operation can model many well-known rearrangements mentioned above, such as reversals, transpositions and block-interchanges, and others not reported in the biological literature. To distinguish those CCLP

operations from the reversal, we call them as non-reversal CCLP operations. Finally, we propose an $O(\delta n)$ time algorithm for solving the weighted sorting problem by CCLP operations when the weight ratio between reversals and non-reversal CCLP operations is 1:2, where n is the number of genes and δ is the number of needed CCLP operations.

Acknowledgement

一轉眼，兩年的研究所生活過去了，我的求學生涯也將告一段落，繼續往人生的下一階段邁進，不管未來的發展如何，都要感謝一路上曾經陪伴、幫助過我的人，特別是我的家人，感謝你們無怨無悔地為我付出，得以讓我完成我的學業。也很感謝我的指導教授，盧錦隆教授，一步一步地指引我的研究方向，讓我可以順利畢業，雖然你常對我們碎碎念，但那些話對我們來說並不刺耳，反而讓我們體會到研究生的本分不只是做好研究，還有很多做人處事的態度及方法值得去學習。

謝謝拖鞋學姐和拖弟，你們的一搭一唱讓我對 genome rearrangements 感到興趣。謝謝忠翰寶貝，讓我知道誰是菇菇界的霸主。謝謝科科以及勞倫濕，讓我在吃肉的旅途上不孤單。謝謝芸蓁，對於從我口中說出的五四三，你都能適時地做出回應。謝謝昱全在 embedded system 的不離不棄以及常常討論對付查某人的策略。謝謝皮老闆和學妹為我跟昱全的口試奔波準備食物，也祝你們的研究順利。

Contents

List of figures

Chapter 1

Introduction

Genome rearrangement studies based on genome-wide analysis of gene orders play an important role in the phylogenetic tree reconstruction [5, 11, 13, 22, 23]. In these studies, a gene is usually represented by a signed integer, where the associated sign indicates on which of the two complementary DNA strands the gene is located, a chromosome by a sequence of genes and a genome by a set of chromosomes. In the last two decades, a variety of rearrangement operations have been proposed to evaluate the evolutionary distance between two related genomes in gene order. Basically, these operations can be classified into two categories: (1) 'intra-chromosomal' rearrangements, such as reversals, transpositions and block-interchanges (also called 'generalized transpositions'), and (2) 'inter-chromosomal' rearrangements, such as fusions, fissions and translocations. *Reversals*, also called *inversions*, affect a segment of consecutive integers in the chromosome by reversing the order and flipping the signs of the integers [2, 13, 14, 17, 24]. *Transpositions* affect two adjacent segments in the chromosome by exchanging their positions [4, 9, 10]. *Block-interchanges* are *generalized transpositions* by allowing the exchanged segments not being adjacent in the chromosome [8, 10, 14,

16, 18]. *Translocations* exchange the end segments between two chromosomes [7, 11, 12, 14, 21]. *Fusions* join two chromosomes into a bigger one and *fissions* break a chromosome into two smaller ones [11, 14, 19, 20].

Recently, the study on the genome rearrangement using block-interchanges has increasingly drawn great attention, since block-interchanges contain transpositions as a special case and, currently, the computational models involving block-interchanges are more tractable than those involving transpositions. More recently, Yancopoulos et al. introduced the double cut and join (DCJ) operation, which cuts the chromosome(s) in two places and rejoins the four cut ends in a new way, as a basis for modeling all the rearrangement operations described previously [25]. Particularly, transpositions and block-interchanges can be modeled by two consecutive DCJ operations, while others by one DCJ operation. In fact, as mentioned in [1], the two consecutive DCJ operations can be viewed as the following procedure to model transpositions or block-interchanges (see Figure 1-1 for a reference). (1) Excision: cut a segment from a chromosome that can be linear or circular. (2) Circularization: join the ends of the excised segment into a temporary circle. (3) Linearization: cut the temporary circle in any place as a linear segment. (4) Reincorporation: paste the linearized segment back to the remaining chromosome at a new site. As also pointed out in [1], this process of fragment excision, circularization, linearization and reincorporation indeed happens in the configuration of the immune response in higher animals. Here, we make a little modification to the reincorporation step in the above process by allowing the linearized segment to be possibly inverted before its reinsertion and also allowing

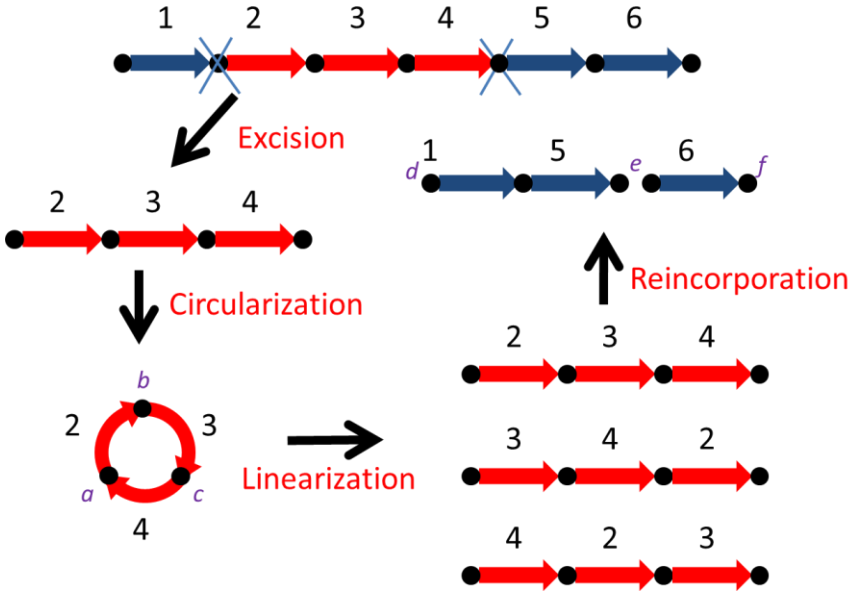

Figure 1-1. (1) Excision: a fragment of genes 2, 3 and 4 is cut from a linear chromosome. (2) Circularization: the two ends of the excised segment are joined together as a circular chromosome that is temporary. (3) Linearization: the temporary circular chromosome is cut at place *a*, *b* or *c* so that it becomes again a linear chromosome. (4) Reincorporation: the linearized chromosome is pasted back to the original chromosome at new site *d*, *e* or *f*.

inverted or non-inverted linearized segment to be pasted back to the remaining chromosome at any site (see Figure 1-2 for the modified model). This modification enables the above cut-circularize-linearize-and -paste (CCLP for short) operation to model seven different kinds of rearrangements, as will be detailed below. It is interesting to note that in addition to transposition and block-interchange, a CCLP operation can model reversal, *inverted transposition* (or *transversal*) [3] and others that are currently not reported in the biological literature. The seven rearrangements modeled by the CCLP operation are described as follows (see Figure 1-2 for a reference).

- Case I – Reversal:

  As illustrated in Figure 1-2, a segment with genes 2, 3 and 4 is cut from a chromosome (1,2,3,4,5,6) and joined as a temporary circle, which is then cut in the same place as it was created by the join (i.e., the *a* site in Figure 1-2), and inverted and pasted back to the chromosome at the cutting site (i.e., the *e* site in Figure 1-2). As a result, this CCLP operation performs as a reversal that changes the chromosome (1,2,3,4,5,6) into (1,-4,-3,-2,5,6).

- Case II – Transposition:

  The temporary circle is cut at a new place (e.g., the *b* site in Figure 1-2) and pasted back to the chromosome at the cutting site. This CCLP operation performs as a transposition that changes (1,2,3,4,5,6) into (1,3,4,2,5,6).

- Case III – Two consecutive, adjacent reversals:

  The temporary circle is cut at a new place (e.g., the *b* site in Figure 1-2), and then inverted and pasted back to the chromosome at the cutting site. This CCLP operation changes (1,2,3,4,5,6) into (1,-2,-4,-3,5,6), which is equivalent to that (1,2,3,4,5,6) is first changed into (1,2,-4,-3,5,6) by a reversal, which is further changed into (1,-2,-4,-3,5,6) by another reversal. Note that the chromosomal regions affected by these two consecutive reversals are adjacent.

- Case IV – Transposition:

  The temporary circle is cut in the same place as it was joined and then pasted back to the chromosome at a new site (e.g., the *f* site in Figure 1-2). This CCLP operation performs as a transposition that

changes (1,2,3,4,5,6) into (1,5,2,3,4,6).

- Case V – Transversal:

  The temporary circle is cut in the same place as it was joined, and then inverted and pasted back to the chromosome at a new site (e.g., the *f* site in Figure 1-2). This CCLP operation performs as an inverted transposition (i.e., transversal) that changes (1,2,3,4,5,6) into (1,5,-4,-3,-2,6).

- Case VI – Block-interchange:

  The temporary circle is cut at a new place (e.g., the *b* site in Figure 1-2) and then pasted back to the chromosome at a new site (e.g., the *f* site in Figure 1-2). This CCLP operation performs as a block-interchange that changes (1,2,3,4,5,6) into (1,5,3,4,2,6).

- Case VII – Two consecutive, overlapping reversals:

  The temporary circle is cut at a new place (e.g., the *b* site in Figure 1-2), and then inverted and pasted back to the chromosome at a new site (e.g., the *f* site in Figure 1-2). This CCLP operation changes (1,2,3,4,5,6) into (1,5,-2,-4,-3,6), which is equivalent to that (1,2,3,4,5,6) is first changed into (1,2,-5,-4,-3,6) by a reversal, which is further changed into (1,5,-2,-4,-3,6) by another reversal. Note that the chromosomal regions affected by these two consecutive reversals are overlapping.

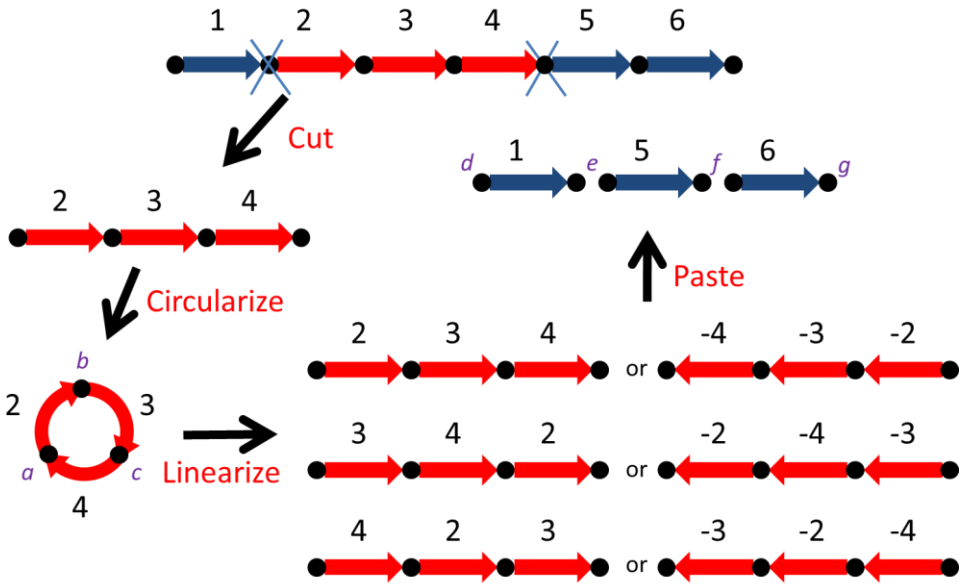

Figure 1-2. A modified cut-circularize-linearize-and-paste operation that can model seven different kinds of rearrangement, where the cutting site of the temporary circle with genes 2, 3 and 4 can be either *a*, *b* or *c*, and the inserting place of the linearized segment at the remaining chromosome can be either *d*, *e*, *f* or *g*.

All these seven rearrangements described above are simply called *CCLP operations*. But, to distinguish those CCLP operations from the reversal, we call them as *non-reversal CCLP operations* in the sequel of this paper. In this article, we are interested in designing efficient algorithms to solve the genome rearrangement problem involving all the seven CCLP operations. If all these CCLP operations are weighted equally, the problem aims to find a minimum number of operations to sort a signed permutation of representing a chromosome. In this case, however, non-reversal CCLP operations are favored in the rearrangement scenario of the optimal solution, as will be clear later, which contradicts with the observation made by biologists that in most organisms, reversals are observed much more frequently when compared with other

rearrangements. Therefore, it may require a reversal to be weighted differently from other CCLP operations. In this differently weighted case, the problem is then called *weighted sorting problem by CCLP operations*, which is to find a sequence of CCLP operations such that the sum of the operation weights in the sequence is minimum. In this study, we focus our attention on the case in which the weight ratio between reversals and non-reversal CCLP operations is 1:2 and use the permutation group in algebra to design an $O(\delta n)$ time algorithm for solving the problem, where n is the number of genes in the given chromosome and δ is the number of needed CCLP operations.

Chapter 2

Preliminaries

Given a set $E = \{1,2,\dots,n\}$ of n positive integer, a *permutation* is a one-to-one mapping from E into itself. For instance, as shown in Figure 2-1, we may define a permutation α of the set $\{1,2,3,4,5,6,7\}$ by $\alpha(1) = 4$, $\alpha(2) = 1$, $\alpha(3) = 6$, $\alpha(4) = 7$, $\alpha(5) = 3$, $\alpha(6) = 5$ and $\alpha(7) = 2$. In the study of genome rearrangement, it is convenient to express the permutation in *cycle* form as $\alpha = (1,4,7,2)(3,6,5)$, in which for each $x \in E$, $\alpha(x)$ is placed directly right to x. A cycle of length k, say $(a_1, a_2, \dots, a_k)$, is simply called *k-cycle* and it also can be written as $(a_i, a_{i+1}, \dots, a_k, a_1, \dots, a_{i-1})$ (i.e., indices are cyclic), where $2 \le i \le k$. Any two cycles are said to be *disjoint* if they have no elements in common. Basically, a permutation can be written in a unique way as the product of disjoint cycles, which is called the cycle decomposition of this permutation, if we ignore the order of the cycles in the product. Usually, a 1-cycle, in which its element is said to be *fixed*, in a permutation is not written explicitly. Especially, the permutation whose elements are all fixed is called an *identity permutation* and is denoted by **1**, i.e., $\mathbf{1} = (1)(2)\cdots(n)$.

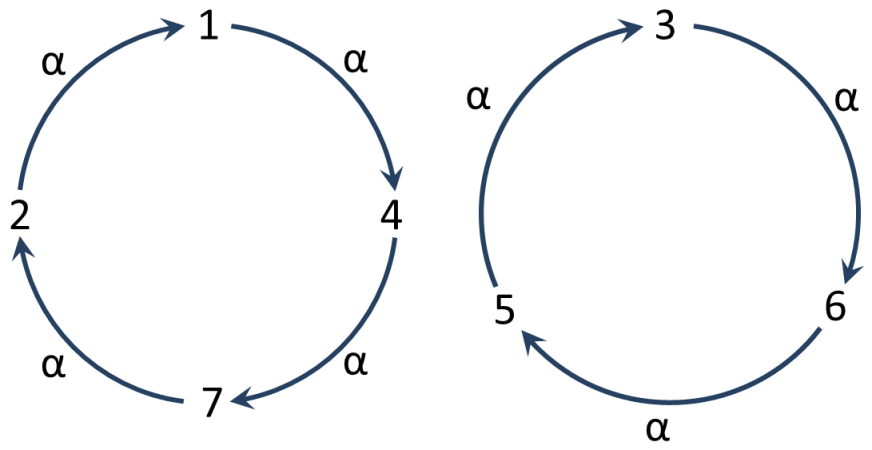

Figure 2-1. The illustration of a permutation $\alpha = (1,4,7,2)(3,6,5)$, where $\alpha(1) = 4$, $\alpha(2) = 1$, $\alpha(3) = 6$, $\alpha(4) = 7$, $\alpha(5) = 3$, $\alpha(6) = 5$ and $\alpha(7) = 2$.

Given two permutations α and β of E, the *composition* (or *product*) of α and β, denote by $\alpha\beta$, is defined to be a permutation of E with $\alpha\beta(x) = \alpha\big(\beta(x)\big)$ for all $x \in E$. For instance, let $\alpha = (2,1)$ and $\beta = (2,5,3,1,6,4)$ be two permutations of $E = \{1,2,3,4,5,6\}$. Then $\alpha\beta = (2,5,3)(1,6,4)$. If α and β are disjoint cycles, then $\alpha\beta = \beta\alpha$. The *inverse* of α is a permutation, denoted as α^{-1}, such that $\alpha\alpha^{-1} = \alpha^{-1}\alpha = \mathbf{1}$. Notably, if a permutation is expressed by the product of disjoint cycles, then its inverse can be obtained by just reversing the order of the elements in each cycle. For example, if $\alpha = (2,5,3)(1,6,4)$, then $\alpha^{-1} = (3,5,2)(4,6,1)$. The *conjugation* of β by α, denoted by $\alpha \cdot \beta$, is the permutation $\alpha\beta\alpha^{-1}$, which is a permutation with the same cycle structure of β but each element x is changed by $\alpha(x)$. More clearly, if $\beta = (b_1, b_2, \dots, b_i)(b_{i+1}, b_{i+2}, \dots, b_k)$, then $\alpha \cdot \beta = (\alpha(b_1), \dots, \alpha(b_i))(\alpha(b_{i+1}), \dots, \alpha(b_k))$.

Let $\alpha = (a_1, a_2)$ be a 2-cycle and β be an arbitrary permutation of *E*. Then the effect of applying α to β can be described as follows:

- If a_1 and a_2 are in the same cycle of β, then this cycle is broken into two smaller ones in $\alpha\beta$ (or $\beta\alpha$), that is, α functions as a *split* operation of β. For instance, $\alpha = (1,2)$ and $\beta = (1,6,4,2,5,3)$, then $\alpha\beta = (1,6,4)(2,5,3)$ and $\beta\alpha = (5,3,1)(6,4,2)$.
- If a_1 and a_2 are in two different cycles of β, then these two cycles are joined into a bigger one in $\alpha\beta$ (or $\beta\alpha$), that is, α functions as a *join* operation of β. For instance, $\alpha = (1,3)$ and $\beta = (1,6,4)(2,5,3)$, then $\alpha\beta = (1,6,4,3,2,5)$ and $\beta\alpha = (6,4,1,2,5,3)$.

Every permutation α of *E* can be expressed as a product of 2-cycles. However, there are many ways of expressing α as a product of 2-cycles. For instance, $(a_1, a_2, \ldots, a_k) = (a_1, a_2)(a_2, a_3)\cdots(a_{k-1}, a_k) = (a_1, a_k)(a_1, a_{k-1})\cdots(a_1, a_2)$, where $k \geq 3$. The *norm* of α, denoted by $\|\alpha\|$, is defined to be the minimum number *k* such that α can be expressed by a product of *k* 2-cycles. Let $n_c(\alpha)$ denote the number of disjoint cycles in the cycle decomposition of α. Notice that $n_c(\alpha)$ also counts the non-expressed 1-cycles. For example, if $\alpha = (1,3,2)(5,6)$ is a permutation of $\{1,2,3,4,5,6\}$, then we have $n_c(\alpha) = 3$, instead of $n_c(\alpha) = 2$, since $\alpha = (1,3,2)(4)(5,6)$. For any permutation α of *E*, it can be shown that $\|\alpha\| = |E| - n_c(\alpha)$ [14, 18]. For two permutations α and β of *E*, α is said to *divide* β, simply denoted by $\alpha|\beta$, if and only if $\|\beta\alpha^{-1}\| = \|\beta\| - \|\alpha\|$. For example, let $\alpha = (2,1)$ and $\beta = (2,5,3,1,6,4)$ be two permutations of $E = \{1,2,3,4,5,6\}$. Then $\beta\alpha^{-1} = (1,5,3)(2,6,4)$. Thus we have $\|\beta\alpha^{-1}\| = 4$, $\|\beta\| = 5$ and

$\|\alpha\| = 1$. As a result, $\|\beta\alpha^{-1}\| = \|\beta\| - \|\alpha\|$ and hence $\alpha|\beta$. Actually, we can easily determine if α divides β using the following lemma.

Lemma 1 [14]. *Let* $a_1, a_2, \ldots, a_k \in E$ *and* β *be any permutation of* E. *Then* $a_1, a_2, \ldots, a_k$ *are in the same cycle of* β *and appear in this cycle in the order of* $a_1, a_2, \ldots, a_k$ *if and only if* $(a_1, a_2, \ldots, a_k)|\beta$.

As mentioned before, a gene is usually represented by a signed integer in the genome rearrangement studies. To properly model a DNA, which is well known as a double stranded molecule, we let $E = \{\pm 1, \pm 2, \ldots, \pm n\}$, in which n is the number of genes and each gene i has counterpart gene $-i$ in the same location in the opposite strand. Let $\Gamma = (1,-1)(2,-2)\cdots(n,-n)$. Clearly, $\Gamma^2 = \mathbf{1}$, that is $\Gamma^{-1} = \Gamma$. A cycle is said to be *admissible* if it does not contain i and $-i$ simultaneously for some gene $i \in E$. Then we can use an admissible n-cycle to represent a DNA strand that is constituted by n genes in some order. Given a DNA strand, say π^+, $\pi^- = \Gamma \cdot (\pi^+)^{-1}$ is its *reverse complement*, since $(\pi^+)^{-1}$ is the reverse of π^+ and $\Gamma \cdot (\pi^+)^{-1}$ is the complement of $(\pi^+)^{-1}$. For our purpose, we here represent the DNA molecule, named π, by the product of the two strands π^+ and π^-, that is, $\pi = \pi^+\pi^- = \pi^-\pi^+$ (since π^+ and π^- are disjoint).

Lemma 2 [14]. *Let* π *and* σ *represent two different chromosomes. If* α *is a cycle in* $\sigma\pi^{-1}$, *then* $(\pi\Gamma)\cdot\alpha^{-1}$ *is also a cycle in* $\sigma\pi^{-1}$.

According to Lemma 2, α and $(\pi\Gamma)\cdot\alpha^{-1}$ are each other's *mate cycle* in $\sigma\pi^{-1}$.

Lemma 3 [14]. *Let u and v be in the different strands of a chromosome* π,

that is, $(u,v) \nmid \pi$. *Then* $\gamma = (\pi\Gamma(v), \pi\Gamma(u))\,(u,v)$ *acts on* π *as a reversal.*

Note that in Lemma 3, (u,v) acts on π as a join operation and $(\pi\Gamma(v), \pi\Gamma(u))$ acts on $(u,v)\pi$ as a split operation. In other words, a reversal acting on π can be implemented by the composition of two 2-cycles and π. In fact, it can be verified that other non-reversal CCLP operations can be implemented by multiplying four 2-cycles $(\pi\Gamma(x), \pi\Gamma(w))(w,x)(\pi\Gamma(v), \pi\Gamma(u))(u,v)$ with the given chromosome π if the following conditions are satisfied: (1) $(u,v)|\pi$, (2) $(w,x) \nmid (u,v)\pi$ (3) $w \neq \Gamma(x)$ or $\Gamma(w) \neq x$ and (4) $(w,\Gamma(x)) \nmid (u,v)\pi$ or $(\Gamma(w),x) \nmid (u,v)\pi$. The first condition is to make sure that (u,v) and $(\pi\Gamma(v), \pi\Gamma(u))$ respectively act on the two strands of π as splits, which lead to two temporary circles excised from π. Note that these two temporary circles are complement to each other. The second condition is to make sure that (w,x) and $(\pi\Gamma(x), \pi\Gamma(w))$ respectively act on the two temporary circles and the cycles of the remaining π as joins, which paste back the two temporary circles into the remaining π. It is worth mentioning that the joins also fulfill the process of linearization with possible inversion. The inversion is performed when the temporary circles are reinserted into the chromosome strands different from the ones they come from. The third and fourth conditions are to make sure that the resulting π are admissible (i.e., no i and $-i$ are in the same chromosome strand). Therefore, we have the following lemma.

Lemma 4. *Let* π *be a chromosome and* $\beta = \big(\pi\Gamma(x), \pi\Gamma(w)\big)(w,x)\big(\pi\Gamma(v), \pi\Gamma(u)\big)(u,v)$. *Suppose that the following four conditions are satisfied:* (1) $(u,v)|\pi$, (2) $(w,x) \nmid (u,v)\pi$ (3) $w \neq \Gamma(x)$ *or* $\Gamma(w) \neq x$ *and* (4) $(w,\Gamma(x)) \nmid (u,v)\pi$ *or*

$(\Gamma(w), x) \nmid (u, v)\pi$. *Then* β *acts on* π *as a non-reversal CCLP operation.*

Chapter 3

Algorithm

In this chapter, we shall utilize the permutation groups to design an efficient algorithm for sorting a given chromosome π into $I = (1,2,\dots,n)(-n,\dots,-2,-1)$ using the CCLP operations when the weight ratio between reversals and non-reversal CCLP operations is 1:2. Recall that any permutation can be expressed as a product of 2-cycles and every reversal (respectively, non-reversal CCLP operation) affecting π can be implemented by a product of two (respectively, four) 2-cycles and π. Furthermore, the composition of $I\pi^{-1}$ and π is *I*, suggesting that $I\pi^{-1}$ can be expressed as a product of 2-cycles that functions as a sequence of CCLP operations to optimally transform π into *I*. In the following, we shall show how to fulfill such an idea. For simplicity, we say that *x* and *y* are *adjacent* in a permutation α if $\alpha(x) = y$ or $\alpha(y) = x$.

Lemma 5. *Let $\pi = \pi^{+}\pi^{-}$ be a chromosome. Suppose that $(x,y)|I\pi^{-1}$ and $(x,y)|\pi$, that is, there are two elements x and y in a cycle of $I\pi^{-1}$ such that (x,y) acts on π as a split. Let $\beta = \big(\pi\Gamma(y), \pi\Gamma(x)\big)(x,y)$. Then there are two adjacent elements x' and y' in a cycle of $I(\beta\pi)^{-1}$ such that (x',y') and $(\beta\pi\Gamma(y'), \beta\pi\Gamma(x'))$ act on $\beta\pi$ as joins. Moreover, the cycles in $\beta'\beta\pi$ are admissible, where*

$\beta' = (\beta\pi\Gamma(y'), \beta\pi\Gamma(x'))(x', y')$.

Proof. For convenience, let $\pi = \pi^+\pi^- = (a_1, a_2, \ldots a_n)(-a_n, -a_{n-1}, \ldots, -a_1)$. The assumption $(x, y)|\pi$ indicates that x and y are in the same cycle of π, say in π^+, and hence $\pi\Gamma(x)$ and $\pi\Gamma(y)$ are in π^-. Hence, both (x, y) and $\big(\pi\Gamma(y), \pi\Gamma(x)\big)$ act on π as splits and $\beta = \big(\pi\Gamma(y), \pi\Gamma(x)\big)(x, y)$ divides π into four cycles. Let $\beta\pi = \pi_1^+\pi_2^+\pi_1^-\pi_2^- = (a_1, \ldots, a_{k-1})(a_k, \ldots, a_n)(-a_{k-1}, \ldots, -a_1)(-a_n, \ldots, -a_k)$. For simplicity of our further discussion, we assume that $a_i < a_{i+1} < n$ for $1 \le i \le k-2$. This indicates that a_{k-1} is the maximum in π_1^+ and hence $a_{k-1}+1$ is not in π_1^+. Moreover, $I(\beta\pi)^{-1}(a_1) = I(a_{k-1}) = a_{k-1}+1$, meaning that a_1 and $a_{k-1}+1$ are adjacent in $I(\beta\pi)^{-1}$. In other words, there are two adjacent elements a_1 and $a_{k-1}+1$ in $I(\beta\pi)^{-1}$ such that $(a_1, a_{k-1}+1)$, as well as $(\beta\pi\Gamma(a_{k-1}+1), \beta\pi\Gamma(a_1))$, acts on $\beta\pi$ as a join. If the two cycles in $(\beta\pi\Gamma(a_{k-1}+1), \beta\pi\Gamma(a_1))(a_1, a_{k-1}+1)\beta\pi$ are admissible (i.e., they represent a chromosome), then we have completed the proof of this lemma based on Lemma 4. Now, suppose that the two cycles in $(\beta\pi\Gamma(a_{k-1}+1), \beta\pi\Gamma(a_1))(a_1, a_{k-1}+1)\beta\pi$ are not admissible (i.e., for some $1 \le i \le n$, both i and $-i$ are in the same cycle). We then show below that we can still find two other adjacent elements x' and y' in a cycle of $I(\beta\pi)^{-1}$ such that (x', y') and $(\beta\pi\Gamma(y'), \beta\pi\Gamma(x'))$ can join $\beta\pi$ into two admissible cycles. First of all, $a_{k-1}+1$ must be in π_1^- (otherwise, $(\beta\pi\Gamma(a_{k-1}+1), \beta\pi\Gamma(a_1))(a_1, a_{k-1}+1)\beta\pi$ is an admissible chromosome), leading to that the cycle created by joining $\pi_1^+\pi_1^-$ using $(a_1, a_{k-1}+1)$ is not admissible. Further suppose that a_j is the minimum in π_1^+. Then $\Gamma(a_j) = -a_j$, which is the maximum in π_1^-. Therefore, we have $-a_j \ge a_{k-1}+1$ (since $a_{k-1}+1$ is also in π_1^-). In

addition, $-a_{j-1}$ and $I(-a_j)$ are adjacent in $I(\beta\pi)^{-1}$ because $I(\beta\pi)^{-1}(-a_{j-1}) = I(-a_j)$. In the following, we consider five possibilities.

Case 1. $a_j \neq -n$ and $a_j \neq 1$. Then $I(-a_j) = -a_j + 1$, which is not in π_1^- since $-a_j$ is the maximum in π_1^-. If $-a_j + 1$ is in π_1^+, then a_{k-1} cannot be the maximum in π_1^+, since $-a_j \geq a_{k-1} + 1$ and hence $-a_j + 1 > a_{k-1}$, which contradicts to our assumption that a_{k-1} is the maximum in π_1^+. In other words, $I(-a_j)$ belongs to either π_2^+ or π_2^- and hence $(-a_{j-1}, I(-a_j))$ acts on $\beta\pi$ as a join and the cycles in $(\beta\pi\Gamma I(-a_j), \beta\pi\Gamma(-a_{j-1}))(-a_{j-1}, I(-a_j))\beta\pi$ are admissible.

Case 2. $a_j = -n$ and both 1 and -1 are not in π_1^+. Then $I(-a_j) = 1$ (instead of $I(-a_j) = -a_j + 1 = n + 1$). Because π_1^+ and π_1^- are complement to each other from chromosomal point of view, both of them contains no 1 and -1, as a result, $I(-a_j)$ belongs to either π_2^+ or π_2^-. Therefore, $(-a_{j-1}, I(-a_j))$ acts on $\beta\pi$ as a join and $(\beta\pi\Gamma I(-a_j), \beta\pi\Gamma(-a_{j-1}))(-a_{j-1}, I(-a_j))\beta\pi$ contains only admissible cycles.

Case 3. $a_j = 1$ and both n and $-n$ are not in π_1^+. Then $I(-a_j) = -n$ (instead of $I(-a_j) = -a_j + 1 = 0$). Clearly, $I(-a_j)$ belongs to either π_2^+ or π_2^-. Therefore, $(-a_{j-1}, I(-a_j))$ acts on $\beta\pi$ as a join and $(\beta\pi\Gamma I(-a_j), \beta\pi\Gamma(-a_{j-1}))(-a_{j-1}, I(-a_j))\beta\pi$ have two admissible cycles.

Case 4. $a_j = -n$ and 1 or -1 is in π_1^+. Because π_1^+ and π_1^- are complement strands, 1 is in π_1^+ if and only if -1 is in π_1^-. Hence, both π_2^+ and π_2^- contains no $-n$, 1 and -1. Then we can exchange the roles of π_1^+ and π_1^- with π_2^+ and π_2^-, respectively, and follow the similar discussion as given in Case 1 to show that we can still find two adjacent elements x' and y' in a cycle of $I(\beta\pi)^{-1}$ such that (x', y') and $(\beta\pi\Gamma(y'), \beta\pi\Gamma(x'))$ can join the four cycles of $\beta\pi$ into two admissible cycles.

Case 5. $a_j = 1$ and n or $-n$ is in π_1^+. Actually, we need not consider this case, because we have initially assumed that all the elements in π_1^+ are less than n and among them, a_j is the smallest.

Based on the above discussion, we have completed the proof of this lemma. ■

Theorem 1. *Let* Φ *denote a minimum weighted sequence of CCLP operations needed to transform* π *into* I. *Then the weight of* Φ *is great than or equal to* $\frac{|E| - n_c(I\pi^{-1})}{2}$.

Proof. Let Φ contain a reversals and b non-reversal CCLP operations. Clearly, the weight of Φ is $a + 2b$. As discussed previously, a reversal can be expressed by a product of two 2-cycles and a non-reversal CCLP operation by a product of four 2-cycles. Therefore, Φ can be written as a product of $2a + 4b$ 2-cycles such that $\Phi\pi = I$, equivalently meaning that $I\pi^{-1}$ can be expressed by a product of $2a + 4b$ 2-cycles and, therefore, $\|I\pi^{-1}\| \leq 2a + 4b$. As mentioned before, based on the lemma proposed in [14,18], we can obtain that $\|I\pi^{-1}\| = |E| - n_c(I\pi^{-1})$. In other words, $|E| - n_c(I\pi^{-1}) \leq 2a + 4b$ and , consequently, the weight

of Φ is great than or equal to $\frac{|E|-n_c(I\pi^{-1})}{2}$. ■

Suppose that there are at least two adjacent elements x and y in a cycle of $I\pi^{-1}$ such that $(x,y)|\pi$. Then, according to Lemma 5, we can always find a non-reversal CCLP operation $\beta'\beta$ from $I\pi^{-1}$ to rearrange π into $\beta'\beta\pi$, where $\beta = \big(\pi\Gamma(y), \pi\Gamma(x)\big)(x,y)$ and $\beta' = (\beta\pi\Gamma(y')$ $,\beta\pi\Gamma(x'))(x',y')$. Suppose that there are no any two adjacent elements x and y in a cycle of $I\pi^{-1}$ such that $(x,y)|\pi$, which implies that $(x,y) \nmid \pi$. Then based on Lemma 3, $\big(\pi\Gamma(y), \pi\Gamma(x)\big)(x,y)$ can serve as a reversal to transform π into $\big(\pi\Gamma(y), \pi\Gamma(x)\big)(x,y)\pi$. Using these properties, we design Algorithm 1 to sort π into *I* by CCLP operations. It is not hard to see that a non-reversal CCLP operation derived in Algorithm 1 decreases the norm of $I\pi^{-1}$ by 4 and a reversal by 2. Since non-reversal CCLP operations are weighted 2 and reversals are weighted 1, Algorithm 1 decreases the norm of $I\pi^{-1}$ by 1 at the weight of $\frac{1}{2}$ and hence its total weight equals to $\frac{\|I\pi^{-1}\|}{2} = \frac{|E|-n_c(I\pi^{-1})}{2}$, which is optimal according to Theorem 1.

Algorithm 1

Input: A chromosome $\pi = (a_1, a_2, \dots, a_n)(-a_n, -a_{n-1}, \dots, -a_1)$.

Output: An optimal scenario Φ of CCLP operations with weight $\omega(\pi, I)$.

1: Compute $I\pi^{-1}$ and $\pi\Gamma$;

2: Let $\omega(\pi, I) = \frac{|E|-n_c(I\pi^{-1})}{2}$ and $\delta = 0$;

3: **while** $\pi \neq I$ **do**

3.1: **if** there exist two adjacent elements x and y in a cycle of $I\pi^{-1}$ such that $(x, y)|\pi$ **then**

3.1.1: Let $\delta = \delta + 1$ and $\beta = \big(\pi\Gamma(y), \pi\Gamma(x)\big)(x, y)$;

3.1.2: Find two adjacent elements x' and y' in a cycle of $I\pi^{-1}\beta$ such that (1) $(x', y') \nmid \beta\pi$, (2) $x' \neq \Gamma(y')$ or $\Gamma(x') \neq y'$ and (3) $(\Gamma(x'), y) \nmid \beta\pi$ or $(x', \Gamma(y')) \nmid \beta\pi$;

3.1.3: Let $\beta' = \big(\beta\pi\Gamma(y'), \beta\pi\Gamma(x')\big)(x', y')$ and $\beta_\delta = \beta'\beta$;

3.1.4: Compute new $\pi = \beta_\delta\pi$ and new $\pi\Gamma = \beta_\delta\pi\Gamma$;

3.1.5: Obtain new $I\pi^{-1}$ by removing y, $\pi\Gamma(x)$, y' and $\beta\pi\Gamma(x')$ from the cycles in original $I\pi^{-1}$;

3.2: **else**

3.2.1: Find two adjacent elements x and y in a cycle of $I\pi^{-1}$ such that $(x, y) \nmid \pi$;

3.2.2: Let $\delta = \delta + 1$ and $\beta_\delta = \big(\pi\Gamma(y), \pi\Gamma(x)\big)(x, y)$;

3.2.3: Compute new $\pi = \beta_\delta\pi$ and new $\pi\Gamma = \beta_\delta\pi\Gamma$;

3.2.4: Obtain new $I\pi^{-1}$ by removing y and $\pi\Gamma(x)$ from the cycles in original $I\pi^{-1}$;

end if

end while

4: Output $\Phi = \beta_1, \beta_2, \ldots, \beta_\delta$ as an optimal scenario with weight $\omega(\pi, I)$;

Theorem 2. *Given a chromosome π, the weighted sorting problem by CCLP operations can be solved in $O(\delta n)$ time when with weight ratio between reversals and non-reversal CCLP operations is* 1:2, *where δ is the number of CCLP operations needed to transform π into I. Moreover, the weight of the optimal solution is $\frac{|E|-n_c(I\pi^{-1})}{2}$ that can be calculated in $O(n)$ time in advance.*

Proof. As discussed previously, Algorithm 1 transforms π into I with a minimum weighted sequence of δ CCLP operations, whose total weight is $\frac{|E|-n_c(I\pi^{-1})}{2}$ that can be calculated in $O(n)$ time. Below, we analyze the time-complexity of Algorithm 1. Basically, steps 1 and 2 can be done in $O(n)$ time. There are δ iterations to perform in step 3. For each iteration of step 3, it takes $O(n)$ time to find (x, y) and (x', y') by determining every pair of adjacent elements in all the cycles of $I\pi^{-1}$ and $I\pi^{-1}\beta$, respectively, and a constant time to perform other operations in step 3.1, and also takes $O(n)$ time to perform step 3.2. Therefore, the total cost of step 3 is $O(\delta n)$. Step 4 is executed in constant time. Totally, the time-complexity of Algorithm 1 is $O(\delta n)$. ■

It should be noted that although the algorithm we presented above takes the circular chromosomes as the instances, it still works for the linear chromosomes because it can be shown that the problem of sorting by CCLP operations is equivalent for circular and linear chromosomes

based on a property, that is, a CCLP operation acting on a gene, say u, on a circular chromosome has an equivalent one that does not act on u (see Figure 4-1 for an example).

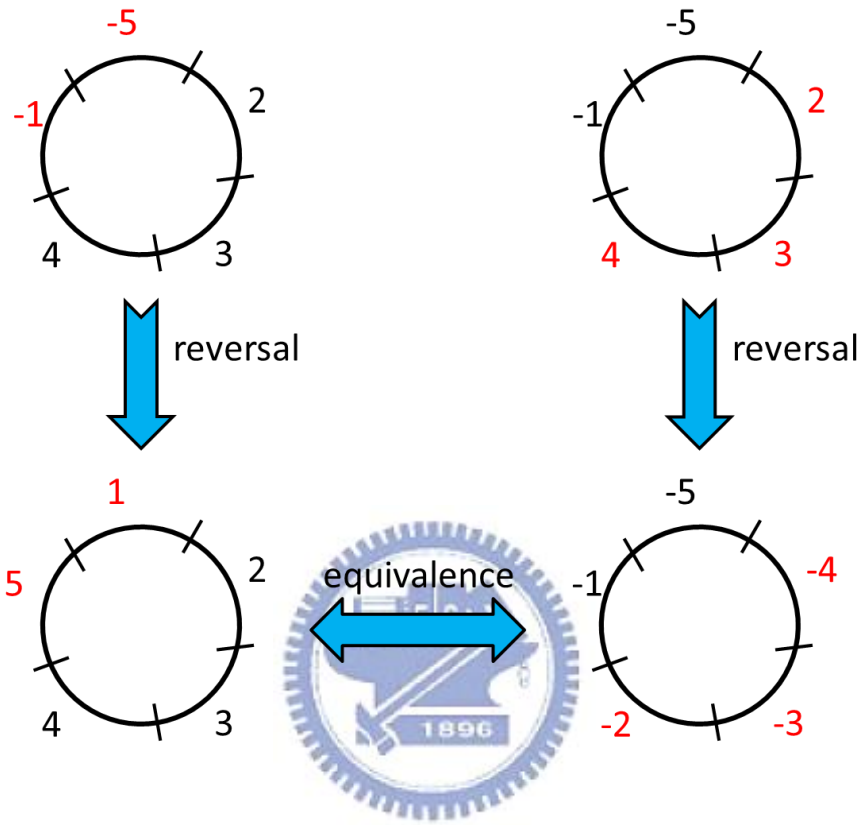

Figure 4-1. A CCLP operation acting on genes -1 and -5 on a circular chromosome has an equivalent one acting on genes 2, 3 and 4.

Chapter 4

Conclusion

In this thesis, we have introduced and studied the sorting problem by CCLP operations, where CCLP is a cut-circularize-linearize-and-paste operation that can model several known and unknown rearrangements. In addition, we have proposed an $O(\delta n)$ time algorithm for solving the weighted sorting problem by CCLP operations when the weight ratio between reversals and non-reversal CCLP operations is 1:2, where *n* is the number of genes and δ is the number of needed CLLP operations. As described in this thesis, this algorithm is very simple so that it can be easily implemented using data structure of 1-dimensional arrays and useful in the studies of phylogenetic tree reconstruction and human immune response to tumors. As a future work, it would be interesting to design efficient algorithms for solving the problem of sorting by CCLP operations when all the CCLP operations are weighted equally.

References

1. Adam, Z. and Sankoff, D., The ABCs of MGR with DCJ, *Evolutionary Bioinformatics*, Vol. 4, pp. 69-74, 2008.

2. Bader, D. A., Moret, B. M. and Yan, M., A linear-time algorithm for computing inversion distance between signed permutations with an experimental study, *Journal of Computational Biology*, Vol. 8, pp. 483-491, 2001.

3. Bader, M. and Ohlebusch, E., Sorting by weighted reversals, transpositions, and inverted transpositions, *Journal of Computational Biology*, Vol. 14, pp. 615-636, 2007.

4. Bafna, V. and Pevzner, P. A., Sorting by transpositions, *SIAM Journal on Discrete Mathematics*, Vol. 11, pp. 221-240, 1998.

5. Belda, E., Moya, A. and Silva, F. J., Genome rearrangement distances and gene order phylogeny in γ-Proteobacteria, *Molecular Biology Evolutionary*, Vol. 22, pp. 1456-1467, 2005.

6. Bergeron, A., Mixtacki, J. and Stoye, J., A unifying view of genome rearrangements, *Lecture Notes in Computer Science*, Vol. 4175, pp. 163-173, 2006.

7. Bergeron, A., Mixtacki, J. and Stoye, J., On sorting by translocations, *Journal of Computational Biology*, Vol. 13, pp. 567-578, 2006.

8. Christie, D. A., Sorting by block-interchanges, *Information Processing Letters*, Vol. 60, pp. 165-169, 1996.

9. Elias, I. and Hartman, T., A 1.375-approximation algorithm for sorting by transpositions, in: Casadio, R. and Myers, G., Eds., *Proceedings of the 5th Work shop on Algorithms in Bioinformatics (WABI 2005)*, *Lecture Notes in Computer Science*, Vol. 3692. Springer-Verlag, pp. 204-215, 2005.

10. Feng, J. X. and Zhu, D. M., Faster algorithms for sorting by transpositions and sorting by block Interchanges, *ACM Transactions on Algorithms*, Vol. 3, No. 3, 2007.

11. Hannenhalli, S. and Pevzner, P. A., Transforming men into mice (polynomial algorithm for genomic distance problem), in: *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science (FOCS 1995)*. IEEE Computer Society, pp. 581-592, 1955.

12. Hannenhalli, S., Polynomial algorithm for computing translocation distance between genomes, *Discrete Applied Mathematics*, Vol. 71, pp. 137-151, 1996.

13. Hannenhalli, S. and Pevzner, P. A., Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals, *Journal of the ACM*, Vol. 46, pp. 1-27, 1999.

14. Huang, Y.-L. and Lu, C. L., Sorting by reversals, generalized block-interchanges, and translocations using permutation groups, *Journal of Computational Biology*, Vol. 17, pp. 685-705, 2010.

15. Huang, Y.-L., Huang, C.-C., Tang, C. Y. and Lu, C. L., SoRT^2: a tool for sorting genomes and reconstructing phylogenetic trees by reversals, generalized transpositions and translocations, *Nucleic Acids Research*, Vol. 38, pp. W221-227, 2010.

16. Huang, Y.-L., Huang, C.-C., Tang, C. Y. and Lu, C. L., An improved algorithm for sorting by block-interchanges based on permutation groups, *Information Processing Letters*, Vol. 110, pp. 345-350, 2010.

17. Kaplan, H., Shamir, R. and Tarjan, R. E., Faster and simpler algorithm for sorting signed permutations by reversals, *SIAM Journal on Computing*, Vol. 29, pp. 880-892, 1999.

18. Lin, Y. C., Lu, C. L., Chang, H.-Y. and Tang, C. Y,. An efficient algorithm for sorting by block-interchanges and its application to the evolution of vibrio species, *Journal of Computational Biology*, Vol. 12, pp. 102-112, 2005.

19. Lu, C. L., Huang, Y.-L., Wang, T. C. and Chiu, H.-T., Analysis of circular genome rearrangement by fusions, fissions and block-interchanges, *BMC Bioinformatics*, Vol. 7, No. 295, 2006.

20. Meidanis, J. and Dias, Z., Genome rearrangements distance by fusion, fission, and transposition is easy, in: Navarro, G., Ed., *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE 2001)*, IEEE Computer Society, pp. 250-253, 2001.

21. Ozery-Flato, M. and Shamir, R., An $O\left(n^{3/2}\sqrt{\log n}\right)$ algorithm for sorting by reciprocal translocations, in: Lewenstein, M. and Valiente, G., Eds., *Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching (CPM 2006)*, *Lecture Notes in Computer Science*, Vol. 4009. Springer, pp. 258-269, 2006.

22. Pevzner, P. and Tesler, G., Genome rearrangements in mammalian evolution: lessons from human and mouse genomes *Genome Research*, Vol. 13, pp. 37-45, 2003.

23. Sankoff, D., Leduc, G., Antoine, N., Paquin, B., Lang, B. F. and Cedergren, R., Gene order comparisons for phylogenetic inference: evolution of the mitochondrial genome, *Proceedings of the National Academy of Sciences*, Vol. 89, pp. 6575-6579, 1992.

24. Tannier, E., Bergeron, A. and Sagot, M.-F., Advances on sorting by reversals, *Discrete Applied Mathematics*, Vol. 155, pp. 881-888, 2007.

25. Yancopoulos, S. Attie, O. and Friedberg, R., Efficient sorting of genomic permutations by translocation, inversion and block-interchanges, *Bioinformatics*, Vol. 21, pp. 3340-3346, 2005.