國 立 交 通 大 學

生物資訊所

碩 士 論 文

利用融合、分裂及區段互換探討基因重組之研究
On the Study of Genome Rearrangements using Fusions, Fissions and Block-Interchanges

研 究 生：王璻菁

指導教授：盧錦隆　教授

中 華 民 國 九 十 四 年 六 月

利用融合、分裂及區段互換探討基因重組之研究

On the study of Genome Rearrangements using Fusions, Fissions and Block-Interchanges

研 究 生：王璻菁　　　　Student：Tsui Ching Wang

指導教授：盧錦隆 教授　　Advisor：Prof. Chin Lung Lu

國 立 交 通 大 學

生 物 資 訊 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

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

中文摘要

最近的研究顯示區段互換的基因體分析可用來衡量兩物種之間的演化距離。所謂的區段互換是一種新的全域型基因體重組, 它的作用是把染色體上兩段不相交且任意長度的基因/標記區段做互換。這方面的研究導致需要去解決一種叫做區段互換排序的組合最佳化數學問題, 其目的是要去找出最少區段互換的次序把一條染色體的基因/標記次序轉成另一條染色體的基因/標記次序。之前, 我們的研究團隊已設計出有效率的演算法可分別解決線性和環狀染色體之間的區段互換排序問題。在本論文中, 我們利用這些演算法實作出一個叫做ROBIN 的網站來分析兩條染色體之間區段互換的基因/標記的次序重組。使用者可輸入兩條或多條線性/環狀且細菌大小的染色體, 輸入的染色體可爲基因體的序列或是代表同源基因或相同標記無正負號的整數序列。爲了測試 ROBIN 的實用性, 我們利用它來偵測三種感染人類的弧菌病原體之間的演化距離關係。結果, 我們的實驗跟先前其它研究團隊利用比較基因體的方法所得到的結果一致, 這項研究成果顯示在弧菌物種間的演化過程中, 區段互換的突變事件扮演著一個重要的角色。然而, 對於含有多條染色體的基因體而言, 它們之間的演化歷史必須還要考慮另外兩種染色體之間的突變事件即融合及分裂。所謂的融合事件是把兩條染色體融合成一條, 而分裂事件是把一條染色體分裂成兩條。在本論文中, 我們進一步的去研究含有區段互換、融合和分裂這三種突變事件的基因體重組問題, 最後提出一個時間複雜度爲$O(n^2)$ 的演算法, 其可有效的計算出兩組環狀且多染色體的基因體之間所需最少上述三種突變次數, 其中n表示兩組基因體之間共有的基因/標記的個數。

Abstract

Analysis of genomes evolving block-interchanges has been studied recently for measuring the evolutionary distance between two organisms, where the *block-interchanges* are new kind of global rearrangement events that affect on a chromosome by swapping two non-intersecting intervals of gene/landmark orders of any length. Such a study leads to a combinatorial problem, called *sorting by block-interchanges*, which is to find to a minimum of block-interchanges needed to transform one chromosome into another. Lin *et al.* [Lin *et al.*, 2005] have designed an efficient algorithm to solve the sorting by block-interchange problem on both cases of linear and circular chromosomes. In this thesis, we implement this algorithm into a web server, called ROBIN, for analyzing rearrangements of gene/landmark orders between two linear/circular chromosomal genomes via the block-interchange events. ROBIN takes two or more linear/circular bacterial-size chromosomes as an input, which can be either genomic sequences or unsigned integer sequences with each representing a homologous gene or identical landmark on all input chromosomes. For testing the applicability of ROBIN, we apply it to genomes of three human *Vibrio* pathogens for detecting the evolutionary relationships. Consequently, our experimental results coincide with the previous ones obtained by other communities using a comparative genomic approach, which implies that the block-interchange events seem to play a significant role in the evolution of *Vibrio* species. However, in the case of genomes consisting of multiple chromosomes, the evolutionary history must also consider chromosomal fusions and fissions, where a *fusion* event merges two chromosomes into one and a *fission* event splits a chromosome into two. Hence, in the thesis, we further study the genome rearrangement problem by fusions, fissions and block-interchanges between two circular multichromosomal genomes. Consequently, we propose an $\mathcal{O}(n^2)$ time algorithm to efficiently compute a minimum series of fusions, fissions and block-interchanges required to transform one genome into another, where n is the number of genes/landmarks shared by the two genomes.

誌 謝

首先, 我要感謝我的指導教授盧錦隆老師耐心的指導, 讓我在碩士兩年期間學到許多做研究應有的態度跟方法; 再來要感謝父母的栽培與支持, 讓我得以無後顧之憂認眞的學習, 還有兄弟姊妹、朋友和實驗室同學、學弟妹的鼓勵, 讓我順利的完成碩士的學業。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With large amounts of various genomic (DNA,RNA and protein sequence) becoming available, genome rearrangements have been studied and proved to be a common model of molecular evolution in mitochondrial, chloroplast, viral, bacterial and mammalian genomes [Hannenhalli & Pevzner, 1999]. In such a study, various rearrangement events, such as reversals (also known as inversions) [Bader *et al.*, 2001, Bafna & Pevzner, 1996, Berman & Hannenhalli, Berman *et al.*, Caprara, 1997, Caprara, 1999, Christie, 1998, Hannenhalli & Pevzner, 1999, Kaplan *et al.*, 2000, Kececioglu & Sankoff, 1993], transpositions [Bafna & Pevzner, 1998, Walter *et al.*, 1998], block-interchanges [Christie, 1996, Lin *et al.*, 2005, Lu *et al.*, 2005], translocations [Hannenhalli, 1996, Kececioglu & Ravi, 1995], fusions, and fissions [Hannenhalli & Pevzner, 1995, Meidanis & Dias, 2001], acting on genes within or among chromosomes have been proposed to determine the evolutionary distance between two related genomes by comparing the gene orders. Almost every genome rearrangement study involves solving the combinatorial optimization problem of finding a series of rearrangements required to transform one genome into another.

Recently, the genome rearrangement study of block-interchanges has increasingly drawn a lot of attention, because block-interchanges may be considered as a generalization of transpositions and currently their computational models measuring the genetic distance are more tractable than those modeled by transpositions.

Christie [Christie, 1996] first introduced the block-interchange events, affecting on a chromosome by swapping two non-intersecting intervals of genes of any length, and proposed an $\mathcal{O}(n^2)$ time algorithm using the breakpoint diagram approach for solving the so-called *block-interchange distance problem* that is to find a minimum series of block-interchanges for transforming one linear chromosome into another, where n is the number of genes. Recently, Lin *et al.* [Lin *et al.*, 2005] have designed a simpler algorithm by making use of the permutation groups in Algebra for solving the block-interchange problem on linear or circular chromosomes with time-complexity of $\mathcal{O}(\delta n)$, where δ is the the minimum number of block-interchanges required for the transformation and can be calculated in $\mathcal{O}(n)$ time in advance. Moreover, they demonstrated that the block-interchange events seem to play a significant role in the evolution of bacterial (*Vibrio*) species.

In this thesis, we first implement a tool, called ROBIN, based on the algorithm designed by Lin *et al.* [Lin *et al.*, 2005] for analyzing rearrangements of gene orders between two linear/circular chromosomal genomes via the block-interchange events. We also integrate an existing program Mauve into our ROBIN so that not only gene-order data but also sequence data are allowed to be the input of ROBIN system. If the input is sequence data, ROBIN can automatically search for the identical homologous/conserved regions shared by all the input sequences. To test the applicability of ROBIN, we apply ROBIN to genomes of three human *Vibrio* pathogens for detecting the evolutionary relationships. Consequently, our experimental results coincide with the previous ones obtained by other community using a comparative genomic approach, which implies that the block-interchange events seem to play a significant role in the evolution of *Vibrio* species.

Notice that the current block-interchanges studies consider only genomes with exactly one chromosome (i.e., unichromosomal genomes) for determining their evolutionary differences. However, for organisms with different numbers of chromosomes, the evolutionary history must also consider chromosome fusions and fissions, where a *fusion* event merges two chromosomes into one and a *fission* event splits a chromosome into two. Hence, it is worthwhile to study the rearrangements of two multichromosomal genomes only based on fusions, fissions and block-interchanges. In this thesis, we consider such a genome rearrangement problem by designing an efficient algorithm for computing a minimum series of fusions, fissions and block-interchanges that are required to transform one multichromosomal genome into another, when both have the same set of genes without repeats. Consequently, we propose an $\mathcal{O}(n^2)$ time algorithm to efficiently compute a minimum series of fusions, fissions and block-interchanges required to transform one genome into another, where n is the number of genes shared by the two genomes.

Chapter 2

Preliminaries

In group theory, a *permutation* is defined to be a one-to-one mapping from a set $E = \{1, 2, \ldots, n\}$ into itself, where n is some positive integer. For example, we may define a permutation α of the set $\{1, 2, 3, 4, 5, 6\}$ by specifying $\alpha(1) = 4, \alpha(2) = 3, \alpha(3) = 1$, $\alpha(4) = 2, \alpha(5) = 7, \alpha(7) = 6$ and $\alpha(6) = 5$. The above mapping can be expressed using a *cycle notation* as illustrated in Figure 2.1 and simply denoted by $\alpha = (1, 4, 2, 3)(5, 7, 6)$. A cycle of length k, say $(a_1, a_2, \ldots, a_k)$, is simply called k-*cycle* and can be rewritten as $(a_i, a_{i+1}, \ldots, a_k, a1, \ldots, a_{i-1})$, where $2 \leq i < k$, or $(a_k, a_1, a_2, \ldots, a_{k-1})$. Any two cycles are said to be *disjoint* if they have no element in common. For any permutation, it 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 [Fraleigh, 1999]. Usually, the cycle of length one of a permutation α is not explicitly written and its element, say x, is said to be *fixed* by α since $\alpha(x) = x$. Especially, the permutation whose elements are all fixed is called an *identity permutation* and is denoted by $\mathbf{1}$ (i.e., $\mathbf{1} = (1)(2)\cdots(n)$).

Given two permutations α and β of E, the *composition* of α and β, denoted by $\alpha\beta$, is defined to be a permutation of E with $\alpha\beta(x) = \alpha(\beta(x))$ for all $x \in E$. If α and β are disjoint cycles, then we have $\alpha\beta = \beta\alpha$. The *inverse* of a permutation α is defined to be a permutation, denoted by α^{-1}, such that $\alpha\alpha^{-1} = \alpha^{-1}\alpha = \mathbf{1}$. 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. Clearly, $\alpha^{-1} = \alpha$ if α is a 2-cycle.

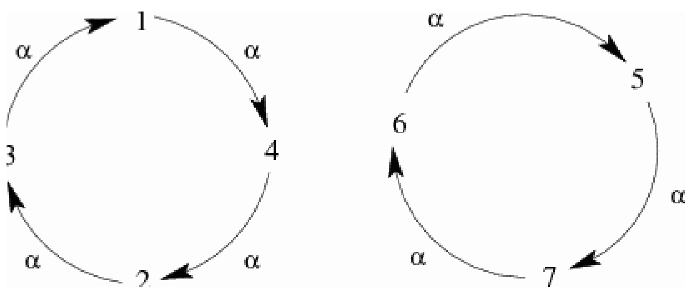


Figure 2.1: The illustration of a permutation $\alpha = (1,4,2,3)(5,6)$ meaning that $\alpha(1) = 4, \alpha(2) = 3, \alpha(3) = 1$, $\alpha(4) = 2, \alpha(5) = 7, \alpha(7) = 6$ and $\alpha(6) = 5$.

Meidanis and Dias [Meidanis & Dias, 2000, Meidanis & Dias, 2001] first noticed that each cycle of a permutation may represent a circular chromosome of a genome with each element of the cycle corresponding to a gene and the order of the cycle corresponding to the gene order of the chromosome. Moreover, they observed that the global evolutionary events like fusions and fissions (respectively, transpositions), correspond to the composition of a 2-cycle (respectively, 3-cycles) and the permutation corresponding to a genome. For instance, given a permutation α whose cycle decomposition is $c_1 c_2 \cdots c_r$. If $\rho = (x, y)$ is a 2-cycle and x and y are in the same cycle of α, say $c_p = (a_1 \equiv x, a_2, \ldots, a_i \equiv y, a_{i+1}, \ldots, a_j)$ where $1 \leq p \leq r$, then in the composition $\rho\alpha$, this cycle c_p is broken into two disjoint cycles $(x \equiv a_1, a_2, \ldots, a_{i-1})$ and $(y \equiv a_i, a_{i+1}, \ldots, a_j)$, i.e., ρ is a fusion event affecting on α (and is called as a *split operation* of α). If $\rho = (x, y)$ is a 2-cycle and x and y are in the different cycles of α, say $c_p = (a_1 \equiv x, a_2, \ldots, a_i)$ and $c_q = (b_1 \equiv y, b_2, \ldots, b_j)$ where $1 \leq p, q \leq r$, then in the composition $\rho\alpha$, c_p and c_q are joined into a cycle $(x \equiv a_1, a_2, \ldots, a_i, y \equiv b_1, b_2, \ldots, b_j)$, i.e., ρ is a fission event affecting on α (and is called as a *join operation* of α). If $\rho = (x, y, z)$ is a 3-cycle and x, y and z are in the same cycle of α, say $c_p = (a_1 \equiv x, a_2, \ldots, a_i, b_1 \equiv y, b_2, \ldots, b_j, c_1 \equiv z, c_2, \ldots, c_k)$ where $1 \leq p \leq r$, then in the composition $\rho\alpha$, this cycle c_p becomes $(x \equiv a_1, a_2, \ldots, a_i, z \equiv c_1, c_2, \ldots, c_k, y \equiv b_1, b_2, \ldots, b_j)$, i.e., ρ is a transposition event affecting on α. It is worth noting that the same results of the above three cases are valid for $\alpha\rho$. Recently, Lin *et al.* [Lin *et al.*, 2005] further observed that a block-interchange event affecting on α corresponds to the composition of two cycles, say ρ_1 and ρ_2, and α under the condition that ρ_1 is a split operation of α and ρ_2 is a join operation of $\rho_1\alpha$. More clearly, let $c_p = (a_1, a_2, \ldots, a_h)$ be a cycle of

α, $\rho_1 = (a_1, a_j)$ and $\rho_2 = (a_i, a_k)$, where $1 < j \leq h$, $1 \leq i \leq j-1$ and $j \leq k \leq h$. Then $\rho_2\rho_1\alpha$ is the resulting permutation by exchanging the blocks $[a_i, a_{j-1}]$ and $[a_k, a_h]$ of c_p.

It is well known that every permutation can be written as a product of 2-cycle. For example, $(1,2,3,4) = (1,4)(1,3)(1,2)$. Actually, there are many ways of expressing a permutation α as a product of 2-cycles [Fraleigh, 1999]. Given a permutation α, let $f(\alpha)$ denote the number of the disjoint cycles in the cycle decomposition of α. Notice that $f(\alpha)$ counts also the non-expressed cycles of length one. For example, if $\alpha = (1,5)(2,4)$ is a permutation of $E = \{1,2,\ldots,5\}$, then $f(\alpha) = 3$, instead of $f(\alpha) = 2$, since $\alpha = (1,5)(2,4)(3)$.

Lemma 1 *Let α be an arbitrary permutation of $E = \{1,2,\ldots,n\}$. If α can be expressed as a product of k 2-cycles, then $k \geq n - f(\alpha)$.*

Proof. Let $\alpha = c_1c_2\ldots c_k$ with each c_i being 2-cycle, where $1 \leq i \leq k$. Next, we prove this lemma by induction on k. Let $k = 0$. Then we have $\alpha = \mathbf{1}$ and hence $f(\alpha) = n$ and $k = n - f(\alpha) = 0$. Suppose that $k - 1 \geq n - f(\alpha')$, where $\alpha' = c1c_2,\ldots,c_{k-1}$. Since $\alpha = \alpha' c_k$ and c_k is a 2-cycle, c_k operates on α' either by joining two cycles of α' into one cycle (i.e., $f(\alpha) = f(\alpha') - 1$) or by splitting one cycle of α' into two cycles (i.e., $f(\alpha) = f(\alpha') + 1$). As a result, we have $f(\alpha) \geq f(\alpha') - 1$ and hence, $k = (k-1) + 1 \geq n - f(\alpha') + 1 = n - (f(\alpha') - 1) \geq n - f(\alpha)$. $\square$

As mentioned previously, each genome with n genes can be expressed by a permutation of $E = \{1,2,\ldots,n\}$. Given two genomes G_1 and G_2 over the same gene set E, the *genome rearrangement distance* from G_1 and G_2, denoted by $d(G_1, G_2)$, is defined to be the minimum number of events needed to transform G_1 into G_2, where the allowed events are fusions, fissions and block-interchanges.

Lemma 2 *Given two genomes G_1 and G_2 over the same gene set E, $d(G_1, G_2) = d(G_2, G_1)$.*

Proof. Let $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_\delta \rangle$ be an optimal series of events required to transform G_1 into G_2. Clearly, $\sigma' = \langle \sigma_\delta, \sigma_{\delta-1}, \ldots, \sigma_1 \rangle$ is an optimal series of events for transforming G_2 into G_1 by reversing the role of every event σ_i, where $1 \leq i \leq \delta$, such that σ_i is a fission (respectively, fusion) in σ' if σ_i is a fusion (respectively, fission) in σ. $\square$

Lemma 3 *Given two genomes G_1 and G_2 over the same gene set E, there is an optimal series of events required to transform G_1 into G_2 such that every fission occurs after every fusion and block-interchange.*

Proof. Let $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_\delta \rangle$ be an optimal series of events needed to transform G_1 into G_2. Of course, if every fission occurs after every fusion and block-interchange in σ, then the proof is done. Now, suppose that not every fission occurs after every fusion or block-interchange in σ. Then let i be the largest index in σ such that σ_i is a fission preceding σ_{i+1} that is a fusion or block-interchange. As discussed below, we can obtain a new optimal series $\sigma' = \langle \sigma_1, \ldots, \sigma_{i-1}, \sigma'_i, \sigma'_{i+1}, \sigma_{i+2}, \ldots, \sigma_\delta \rangle$ to transform G_1 into G_2 such that σ'_i is a fusion or block-interchange and σ'_{i+1} is a fission. Suppose that σ_i splits a chromosome A into A_1 and A_2. If σ_{i+1} is a fusion, then we assume that it joins two chromosomes B_1 and B_2 into B; otherwise, if σ_{i+1} is a block-interchange, then assume that it affects B_1 such that B_1 becomes B through a block-interchange. Clearly, if neither B_1 nor B_2 is created by σ_i, then the desired series σ' is obtained by swapping σ_i and σ_{i+1} in σ (i.e., $\sigma'_i = \sigma_{i+1}$ and $\sigma'_{i+1} = \sigma_i$). If both B_1 and B_2 are created by σ_i, then by removing σ_i and σ_{i+1} from σ, we obtain a new optimal series of events transforming G_1 into G_2 with the smaller number of events than δ, a contradiction. Hence, we assume that only one of B_1 and B_2 is created by σ_i and without loss of generality, let $B_1 = A_1$. Now, consider the following two cases.

Case 1: σ_{i+1} is a fusion. (Suppose the considered chromosomes are circular.) Then the net rearrangement caused by σ_i and σ_{i+1} is to transform A and B_2 into A_2 and B. Actually, this rearrangement can also be done by first joining A and B_2 via σ_{i+1} and then splitting it into A_2 and B via σ_i. Then σ' is obtained by just swapping σ_i and σ_{i+1} in σ.

Case 2: σ_{i+1} is a block-interchange. Clearly, the net rearrangement caused by σ_i and σ_{i+1} is to transform A into B and A_2, which is equivalent to the rearrangement by first applying σ_{i+1} to A and then further splitting it into B and A_2 via σ_i. Then σ' is obtained by swapping σ_i and σ_{i+1} in σ.

According to the discussion above, we can always obtain σ' by simply swapping σ_i and σ_{i+1} in σ. Repeating this way on the resulting σ', we can finally obtain an optimal series of events that are required to transform G_1 into G_2 such that all fissions come after all fusions and block-interchanges. $\square$

Lemma 4 *Given two genomes G_1 and G_2 over the same gene set E, there is an optimal series of events required to transform G_1 into G_2 such that all fusions come before all block-interchanges which come before all fissions.*

Proof. Let $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_\delta \rangle$ be an optimal series of events required to transform G_1 into G_2. If there are no fusions or block-interchanges, then the proof is completed. If not, according to Lemma 3, we may assume that all fusions and block-interchanges occur before all fissions. Let i be the index of the last non-fission in σ and also let G' be the resulting genome after all $\sigma_1, \sigma_2, \ldots, \sigma_i$ have affected G_1. Since σ is optimal, it is not hard to see that $\sigma' = \langle \sigma_1, \sigma_2, \ldots, \sigma_i \rangle$ is an optimal series of fusions and block-interchanges needed to transform G_1 into G'. As discussion in the proof of Lemma 2, $\sigma'' = \langle \sigma_i, \sigma_{i-1}, \ldots, \sigma_1 \rangle$ is an optimal series of fissions and block-interchanges for transforming G' into G_1 and by Lemma 3, all these i fissions and block-interchanges in σ'' can be rearranged such that all block-interchanges occur before all fissions. In other words, the i fusions and block-interchanges in σ' can be rearranged such that all fusions occur before all block-interchanges. Consequently, there is an optimal series of events needed to transform G_1 into G_2 such that all fusions come before all block-interchanges which come before all fissions. $\square$

Chapter 3

ROBIN: A Tool for Genome Rearrangements of Block-Interchanges

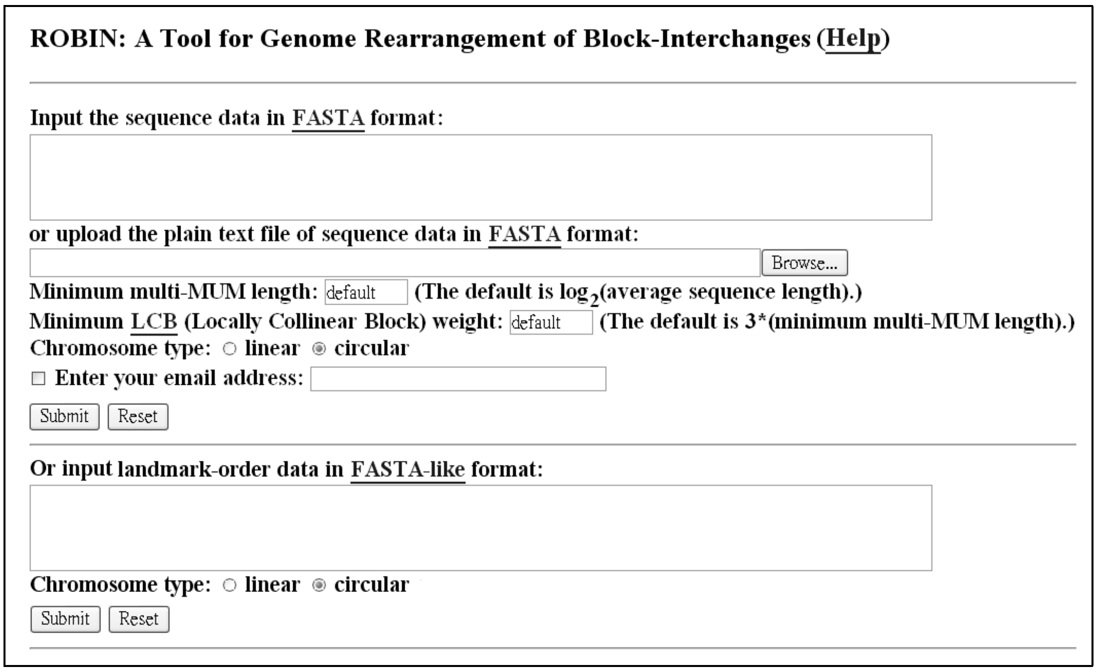
In this chapter, we implement the algorithm designed by Lin *et al.* [Lin *et al.*, 2005] into a web server, called ROBIN, for analyzing genome rearrangement of block-interchanges between two chromosomal genomes. It takes two or more linear/circular chromosomes as its input, and computes the number of minimum block-interchange rearrangements between any two input chromosomes for transforming one chromosome into another and also determines an optimal scenario taking this number of rearrangements. The input can be either bacterial-size sequence data or landmark-order data. If the input is sequence data, ROBIN will automatically search for the identical landmarks that are the homologous/conserved regions shared by all the input sequences.

3.1 Introduction

With the increasing number of sequenced genomes, the study of genome rearrangement, which measures the evolutionary difference between two organisms by conducting a large-scale comparisons of their genomic data, has received a lot of attention in computational biology and bioinformatics. One of the most promising ways to do this research is to compare the orders of the identical landmarks in two different genomes, where the identical landmarks can be the homologous/conserved regions (including genes) shared by the sequences. The genomes considered are usually denoted by a set of ordered (signed or unsigned) integers with each integer representing an identical landmark in the genomes and its sign ($+$ or $-$) indicating the transcriptional orientation. Given a set of ordered landmarks from each genome, many existing tools [Tesler, 2002, Pevzner & Tesler, 2003, Darling *et al.*, 2004*b*] have focused on inferring an optimal series of reversal events that transform one genome organization into another, where the reversal events act on the genome by inverting a contiguous interval of landmarks into the reverse order and also inverting the orientation of each landmark. Other rearrangements like transpositions [Bafna & Pevzner, 1998, Walter *et al.*, 1998], translocations [Hannenhalli, 1996, Kececioglu & Ravi, 1995], fissions, fusions [Hannenhalli & Pevzner, 1995, Meidanis & Dias, 2001] and block-interchanges [Christie, 1996, Lin *et al.*, 2005] have been proposed to determine the evolutionary distance between two related genomes. Christie [Christie, 1996] first introduced the block-interchange events, a new kind of global rearrangements affecting on a genome by swapping two non-intersecting intervals of landmarks of any length, and proposed an $\mathcal{O}(n^2)$ time algorithm for solving the so-called *block-interchange distance problem* that is to find a minimum series of block-interchanges for transforming one linear genome into another, where n is the number of landmarks. In fact, the block-interchanges can be considered as a generalization of the transpositions because the intervals of landmarks swapped by a block-interchange event are not necessarily adjacent. Recently, Lin *et al.* [Lin *et al.*, 2005] have designed a simpler algorithm for solving the block-interchange problem on linear or circular genomes with time-complexity of $\mathcal{O}(\delta n)$, where δ is the minimum number of block-interchanges required for the transformation and can be calculated in $\mathcal{O}(n)$ time in advance. They also demonstrated that the block-interchange events seem to play a significant role in the evolution of bacterial (vibrio) species. Actually, the proof in the paper of Lin *et al.* [Lin *et al.*, 2005] for showing their circular algorithm being able to apply to linear chromosomes can be easily extended to prove that the block-interchange problem on circular genomes is equivalent to that on linear genomes. Here, we adopt their algorithms to implement the kernel of ROBIN (a short for Rearrangement Of Block-INterchanges) program for analyzing rearrangements of landmark orders between two linear/circular chromosomal genomes via the block-interchange events. In addition, by integrating Mauve [Darling *et al.*, 2004*a*] into our ROBIN system, not only landmark-order data but also sequence data are allowed to be the input of ROBIN system. If the input is sequence data, ROBIN can automatically search for the identical landmarks that are the homologous/conserved regions shared by all the input sequences.

3.2 Method

For the landmarks used in the analyses of rearrangement among genomes, we considered the exact matches such as the maximal unique matches (MUMs) as in MUMmer [Delcher *et al.*, 1999, Delcher *et al.*, 2002], the approximate matches without gaps such as the yielding fragments as in DIALIGN [Morgenstern *et al.*, 1998, Morgenstern, 1999] and LAGAN [Brudno *et al.*, 2003], the approximate matches with gaps such as the hit fragments as in BLASTZ [Schwartz *et al.*, 2003] or the regions of local collinearity such as the locally collinear blocks (LCBs) as in Mauve [Darling *et al.*, 2004*a*]. Conceptually, an LCB can be considered as a collinear (consistent) set of the multi-MUMs, where multi-MUMs are exactly matching subsequences shared by all the considered genomes that occur only once in each of genomes and that are bounded on either side by mismatched nucleotides. Here, we adopt the LCBs for representing the landmarks in genomes. The main reason is that each LCB may correspond to a homologous region of sequence shared by all genomes and does not contain any genome rearrangements. In addition, Darling *et al.* [Darling *et al.*, 2004*a*] have implemented a package called Mauve that contains a program which is able to efficiently identify all the LCBs shared by all the large-scale genomes being studied. Usually, each identified LCB is associated with a weight that can serve as a measure of confidence that it is a true homologous region rather than a random match, where the weight of an LCB is defined as the sum of the lengths of multi-MUMs in this LCB. By selecting a high minimum weight, the user


ROBIN: A Tool for Genome Rearrangement of Block-Interchanges (Help)

Input the sequence data in FASTA format:

or upload the plain text file of sequence data in FASTA format:

Browse...

Minimum multi-MUM length: default **(The default is $\log_2$(average sequence length).)**

Minimum LCB (Locally Collinear Block) weight: default **(The default is 3*(minimum multi-MUM length).)**

Chromosome type: linear [✓ circular]

☐ **Enter your email address:**

Submit Reset

Or input landmark-order data in FASTA-like format:

Chromosome type: linear [✓ circular]

Submit Reset

Figure 3.1: The web interface of ROBIN.

can identify the larger LCBs that are truly involved in genome rearrangement, whereas by selecting a low minimum weight, the user can trade some specificity for sensitivity to identify the smaller LCBs that are possibly involved in genome rearrangement. For the detailed algorithm of computing LCBs, we refer the reader to the paper by Darling *et al.* [Darling *et al.*, 2004*a*, Darling *et al.*, 2004*b*].

The kernel algorithms of ROBIN are written in C++ and the web interface is written in PHP. It can be easily accessed via a simple web interface (see Figure 3.1). The input of ROBIN can be two or more linear/circular chromosomes with bacterial size that can be either genomic sequences or unsigned integer sequences with each integer representing an identical landmark on all input chromosomes. If the input is genomic sequences, our ROBIN will automatically identify all the LCBs (i.e., homologous/conserved regions) that meet the user-specified minimum weight, where the minimum LCB weight is a user-definable parameter and our ROBIN chooses its default to be 3 times the minimum multi-MUM length. The output of ROBIN is the block-interchange distance between any two input chromosomes and its optimal scenario of block-interchange rearrangements for transforming one chromosome into another.

ROBIN is freely accessed at http://genome.life.nctu.edu.tw/ROBIN.

Table 3.1: The running time of ROBIN for processing two landmark orders with different size.

Number of Landmarks	CPU Time Usage
100	$<$ 1 sec
1000	55 sec
2000	3.7 min
3000	9 min
4000	28 min

Table 3.2: The running time of ROBIN for processing multiple landmark orders of 1,000 landmarks.

Number of Landmark Orders	CPU Time Usage
2	51 sec
3	144 sec
4	289 sec
5	472 sec

3.3 CPU Time Usage and Experimental Results

3.3.1 CPU Time Usage

Currently, the ROBIN system is installed on IBM PC with 1.26 GHz processor and 512 MB RAM under Linux system. On such hard environment, our ROBIN can deal with only about 4,800 landmarks if the input is landmark-order data. The running time of ROBIN for dealing with two landmark orders is shown in Table 3.1 and Table 3.2 where Table 3.1 shows the running time of ROBIN for processing two landmark orders when the number of landmarks is increased, and the table 3.2 shows the running time of ROBIN for processing multiple landmark orders with 1,000 landmarks when the number of landmark orders is increased.

If the input is sequence data, the limitation of our ROBIN greatly depends on the length scale and number of input sequences, because it needs additional time for computing all the LCBs shared by all input sequences. Currently, it can handle the sequences with total length of up to 35 Mbp. The following tables list the running time of ROBIN for processing two sequences when the average length of sequences is increased, where Table 3.3 shows the running time of ROBIN for processing sequence

Table 3.3: The running time of ROBIN for processing two sequences with different length.

Average Sequence Length(Mbp)	CPU Time Usage
1.9	15 min
3.4	18 min
5	46 min
8	56 min

Table 3.4: The running time of ROBIN for processing multiple sequences.

Number of Sequences	CPU Time Usage
2	24 min
3	53 min
4	81 min
5	157 min
6	240 min
7	290 min

data of length ranging from 4.6 Mbp to 5.5 Mbp when the number of sequences is increased, and Table 3.4 shows the running time of ROBIN for processing sequence data of length ranging from 4.6 Mbp to 5.5 Mbp when the number of sequences is increased.

3.3.2 Experimental Results

To test our ROBIN system, we rerun the experiments conducted by Lin et al. [Lin *et al.*, 2005], for detecting the evolutionary relationships among three human *Vibrio* pathogens, including *V. vulnificus*, *V. parahaemolyticus* and *V. cholerae*(see the Table 3.5 for their sequence information). It is reported that *V. vulnificus* is an etiologic agent for severe human infection acquired through wounds or contaminated seafood and shares morphological and biochemical characteristics with other human vibrio pathogens, including *V. cholerae* and *V. parahaemolyticus* [Chen *et al.*, 2003]. The genomes of these three vibrio species consist of two circular chromosomes, and their genomic sequences have been uncovered recently. As more and more sequence information of *Vibrio* species becomes available, a comparative genomics approach is needed

Table 3.5: The sequence information of three pathogenic *Vibrio* Species, each with two circular chromosomes.

Accession NO	Species	Chromosome Size	(Mbp)
NC_005139	*V. vulnificus* YJ016	1 (VV1)	3.4
NC_005140	*V. vulnificus* YJ016	2 (VV2)	1.9
NC_004603	*V. parahaemolyticus* RIMD 2210633	1 (VP1)	3.3
NC_004605	*V. parahaemolyticus RIMD* 2210633	2 (VP2)	1.9
NC_002505	*V. cholerae* El Tor N16961	1 (VC1)	3.0
NC_002506	*V. cholerae* El Tor N16961	2 (VC2)	1.0

to uncover the critical events leading to the functional uniqueness of *Vibrio* species. To address the issue of how vibrio species evolved, Chen et al. [Chen *et al.*, 2003] conducted a chromosome-by-chromosome analysis of the *V. vulnificus* YJ016 sequence along with the *V. cholerae* El Tor N16961 sequence and the *V. parahaemolyticus* RIMD 2210633 sequence to compare relative positions of conserved genes and to investigate the movement of genetic materials within and between the two chromosomes in the vibrio species. Their comparative analysis revealed that *V. vulnificus* showed a higher degree of conservation in gene organization in the two chromosomes relative to *V. parahaemolyticus* than to *V. cholerae*, which implies that *V. vulnificus* is closer to *V. parahaemolyticus* than to *V. cholerae* from the evolutionary viewpoint. Chen et al. [Chen *et al.*, 2003] also conducted an analysis by comparing the number, distribution, and position of gene family members in the *V. vulnificus* and *V. cholerae* genomes. The results indicated that it appears that duplication and transposition events occurred more frequently in the *V. vulnificus* genome. Since the transposition is a special case of block-interchange, it seems to be reasonable to postulate that the rearrangement of block-interchange may play another significant role in the evolution of *Vibrio* genomes. To justify this viewpoint, we conducted an experiment on these three human *Vibrio* pathogens to see if their evolutionary relationships determined only based on their block-interchange distances with each other agree with those obtained by Chen et al. [Chen *et al.*, 2003].

In the previous experiments as we have done in [Lin *et al.*, 2005], we used the common MUMs, which were computed in advance with another tool of finding consensuses

Table 3.6: The block-interchange distances among VV1, VP1, and VC1.

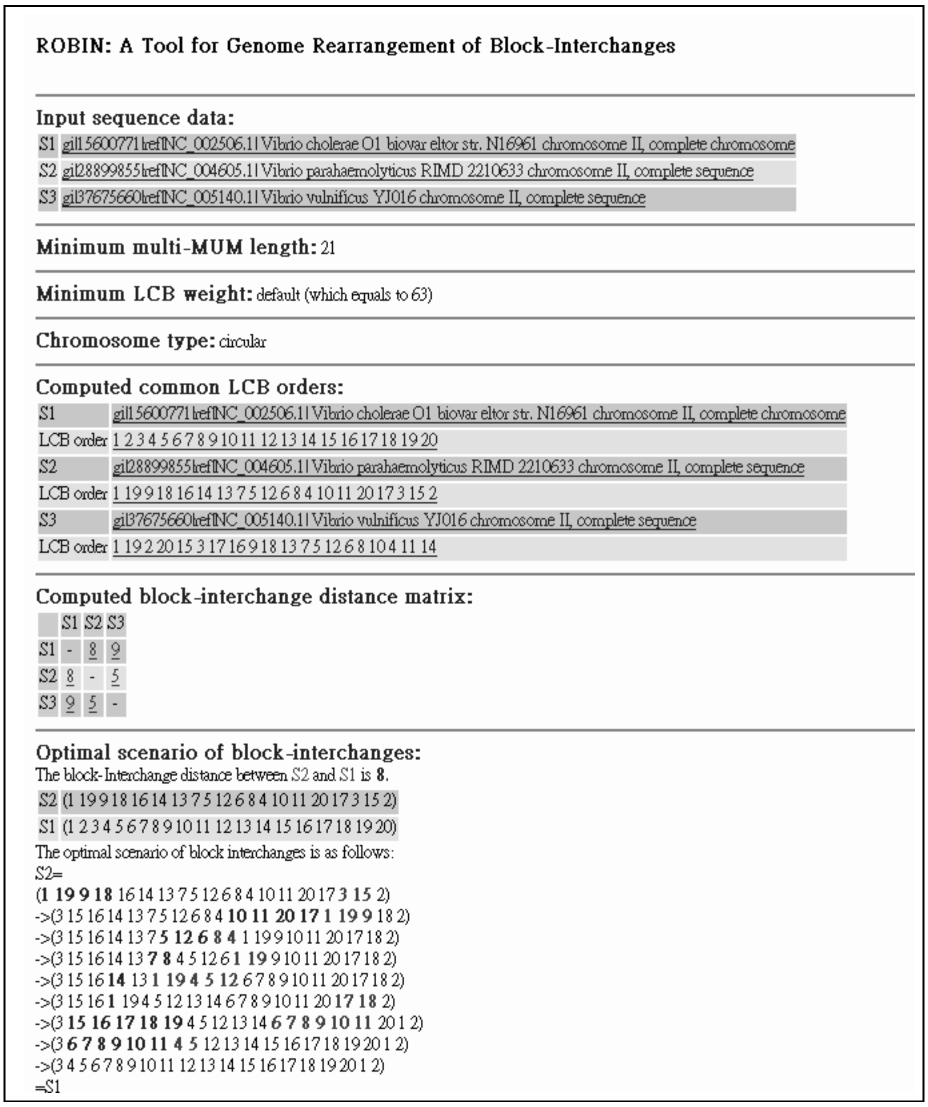
	VC1	VP1	VV1
VC1	-	37	38
VP1	37	-	17
VV1	38	17	-

Table 3.7: The block-interchange distances among among VV2, VP2, and VC2.

	VC2	VP2	VV2
VC2	-	8	9
VP2	8	-	5
VV2	9	5	-

or signatures, among these three vibrio genomes to represent the identical landmarks. However, in the experiments we have done here, we used the LCBs as the landmarks that were automatically computed by our ROBIN system with default parameters from three input *Vibrio* genomic sequences.

Totally, ROBIN identified 95 (respectively, 20) common LCBs for VV1, VP1, and VC1 (respectively, VV2, VP2, and VC2). The computed block-interchange distance matrices are shown in Table~̃reftab:5 and Table~̃reftab:6 . Table 3.7 As shown in the table above, the block-interchange distance between *V. vulnificus* and *V. parahaemolyticus* is smaller than that between *V. vulnificus* and *V. cholerae* and that between *V. parahaemolyticus* and *V. cholerae*in both circular chromosomes. These experimental results indeed coincide with those obtained by Lin et al. [Lin *et al.*, 2005] and by Chen et al. [Chen *et al.*, 2003].



ROBIN: A Tool for Genome Rearrangement of Block-Interchanges

Input sequence data:

S1	gi|15600771|ref|NC_002506.1| Vibrio cholerae O1 biovar eltor str. N16961 chromosome II, complete chromosome
S2	gi|28899855|ref|NC_004605.1| Vibrio parahaemolyticus RIMD 2210633 chromosome II, complete sequence
S3	gi|37675660|ref|NC_005140.1| Vibrio vulnificus YJ016 chromosome II, complete sequence

Minimum multi-MUM length: 21

Minimum LCB weight: default (which equals to 63)

Chromosome type: circular

Computed common LCB orders:

S1	gi|15600771|ref|NC_002506.1| Vibrio cholerae O1 biovar eltor str. N16961 chromosome II, complete chromosome
LCB order	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
S2	gi|28899855|ref|NC_004605.1| Vibrio parahaemolyticus RIMD 2210633 chromosome II, complete sequence
LCB order	1 19 9 18 16 14 13 7 5 12 6 8 4 10 11 20 17 3 15 2
S3	gi|37675660|ref|NC_005140.1| Vibrio vulnificus YJ016 chromosome II, complete sequence
LCB order	1 19 2 20 15 3 17 16 9 18 13 7 5 12 6 8 10 4 11 14

Computed block-interchange distance matrix:

	S1	S2	S3
S1	-	8	9
S2	8	-	5
S3	9	5	-

Optimal scenario of block-interchanges:

The block-Interchange distance between *S2* and *S1* is **8**.

S2	(1 19 9 18 16 14 13 7 5 12 6 8 4 10 11 20 17 3 15 2)
S1	(1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)

The optimal scenario of block interchanges is as follows:

S2=
(**1 19 9 18** 16 14 13 7 5 12 6 8 4 10 11 20 17 **3 15** 2)
->(3 15 16 14 13 7 5 12 6 8 4 **10 11 20 17 1 19 9** 18 2)
->(3 15 16 14 13 7 **5 12 6 8 4** 1 19 9 10 11 20 17 18 2)
->(3 15 16 14 13 **7 8** 4 5 12 6 **1 19** 9 10 11 20 17 18 2)
->(3 15 16 **14** 13 **1 19 4 5 12** 6 7 8 9 10 11 20 17 18 2)
->(3 15 16 **1** 19 4 5 12 13 14 6 7 8 9 10 11 20 **17 18** 2)
->(3 **15 16 17 18 19** 4 5 12 13 14 **6 7 8 9 10 11** 20 1 2)
->(3 **6 7 8 9 10 11 4 5** 12 13 14 15 16 17 18 19 20 1 2)
->(3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 1 2)
=*S1*

Figure 3.2: The partial result of ROBIN.

Chapter 4

Algorithm of Genome Rearrangements by Fusions, Fissions and Block-Interchanges

Let α and I be two given multichromosomal genomes over the same gene set $E = \{1, 2, \ldots, n\}$. Here, we assume that the genes in I are sorted in a way that gene $i+1$ is on the right of gene i if gene $i+1$ is not an ending gene on the left of a chromosome; otherwise, gene i is an ending gene on the right of another chromosome, where $1 \leq i \leq n-1$. For example, $I = (1,2)(3,4,5)(6,7,8,9)$. In this case, the computation of $d(\alpha, I)$ and its corresponding optimal scenario can be considered as a problem of *sorting* α using the minimum fusions, fissions and block-interchanges. Recall that any series of fusions, fissions and block-interchanges for transforming α into I can be expressed by a product of 2-cycles, say $c_\lambda c_{\lambda-1} \ldots c_1$, such that $c_\lambda c_{\lambda-1} \ldots c_1 \alpha = I$ and hence $c_\lambda c_{\lambda-1} \ldots c_1 = I\alpha^{-1}$. This property implies that $I\alpha^{-1}$ contains all information that can be utilized to derive $c_1, c_2, \ldots, c_\lambda$ for transforming α into I. Based on this observation and Lemma 4, we design below an efficient algorithm for computing $d(\alpha, I)$ and its optimal scenario of rearrangement events.

Given a multichromosomal genome α, we denote by $\chi(\alpha)$ the number of chromosomes in α. For any two multichromosomal genomes $\alpha = \alpha_1\alpha_2 \ldots \alpha_{\chi(\alpha)}$ and $I = I_1 I_2 \ldots I_{\chi(I)}$ over the gene set $E = \{1, 2, \ldots, n\}$, an undirected graph $\mathcal{G}(\alpha, I) =$

$(\mathcal{V}_\alpha, \mathcal{V}_I, \mathcal{E})$ is constructed from α and I as follows.

- $\mathcal{V}_\alpha = \{\alpha_1, \alpha_2, \alpha_{\chi(\alpha)}\}$.
- $\mathcal{V}_I = \{I_1, I_2, I_{\chi(I)}\}$.
- $\mathcal{E} = \{(\alpha_i, I_j) | 1 \leq i \leq \chi(\alpha), 1 \leq j \leq \chi(I)$, and α_i and I_j have at least a common gene $\}$.

It is not hard to see that $\mathcal{G}(\alpha, I)$ is a bipartite graph since $\mathcal{V}_\alpha$ and $\mathcal{V}_I$ are independent sets (i.e., no edge between any two vertices in $\mathcal{V}_\alpha$ or $\mathcal{V}_I$) in $\mathcal{G}(\alpha, I)$. If there are two genes in a chromosome I_k of I that appear in two different chromosomes α_i and α_j of α, then both α_i and α_j belong to the same connected component in $\mathcal{G}(\alpha, I)$ since $(\alpha_i, I_k) \in \mathcal{E}$ and $(\alpha_j, I_k) \in \mathcal{E}$, where the *connected component* is defined to be a maximal subgraph of $\mathcal{G}(\alpha, I)$ such that there exists a path between any pair of vertices in this subgraph. Let $\mathsf{comp}(\mathcal{G}) = \{P_1, P_2, \ldots, P_\omega\}$ denote the collection of all connected components in $\mathcal{G}(\alpha, I)$. For each $P_i \in \mathsf{comp}(\mathcal{G})$, let $\mathsf{genome}(P_i, \alpha)$ (respectively, $\mathsf{genome}(P_i, I)$) denote the set of chromosomes whose corresponding vertices are in $\mathcal{V}_\alpha$ (respectively, $\mathcal{V}_I$) and let $\mathsf{gene}(P_i, \alpha)$ (respectively, $\mathsf{gene}(P_i, I)$) denote the union of the genes in all the chromosomes of $\mathsf{genome}(P_i, \alpha)$ (respectively, $\mathsf{genome}(P_i, I)$). Note that $\mathsf{gene}(P_i, \alpha) = \mathsf{gene}(P_i, I)$. By Lemma 4, there is always an optimal series of events, say σ, required to transform α into I in which all fusions precede all block-interchanges that further precede all fissions. Let n_{fu}, n_{bi} and n_{fi} denote the numbers of fusions, block-interchanges and fissions in σ, respectively. Notice that the chromosomes considered here are disjoint (i.e., without gene duplication). Hence, for any two chromosomes α_i and α_j whose corresponding vertices are in the same connected component of $\mathcal{G}(\alpha, I)$, there must exist a fusion in σ that joins α_i and α_j into one chromosome. In addition, we have $n_{fu} = \sum_{1 \leq i \leq \omega}(|\mathsf{genome}(P_i, \alpha)| - 1)$, since all needed fusions come together in the beginning of σ. After these n_{fu} fusions, the resulting α, denoted by α', has ω chromosomes, say $C_1, C_2, \ldots, C_\omega$, with each C_i consisting of the genes in $\mathsf{gene}(P_i, \alpha)$. Due to being intra-chromosomal events, the next n_{bi} block-interchanges come before all fissions to mutate α' without changing the gene content of any chromosome in α'. As a result, $n_{fi} = \sum_{1 \leq i \leq \omega}(|\mathsf{genome}(P_i, I)| - 1)$, since $\mathsf{gene}(P_i, \alpha) = \mathsf{gene}(P_i, I)$ for each

$1 \le i \le \omega$ and the chromosomes in I are disjoint. Hence, we have the following lemma immediately.

Lemma 5 *Given two multichromosomal genomes α and I over the same gene set E, let $\{P_1, P_2, \ldots, P_\omega\}$ be the collection of all connected components in the induced bipartite graph $\mathcal{G}(\alpha, I)$. Then there is an optimal series of events for transforming α into I in which the number of fusions is $n_{fu} = \sum_{1\le i\le\omega}(|\mathsf{genome}(P_i, \alpha)| - 1)$ and the number of fissions is $n_{fi} = \sum_{1\le i\le\omega}(|\mathsf{genome}(P_i, I)| - 1)$.*

Actually, in the discussion above, we can derive σ' from σ by rearranging the events in σ in a way that all the events, which mutate the chromosomes originated from some initial chromosomes in α, come together in σ' such that all fusions precede all block-interchanges which precede all fissions. That is, we can solve the genome rearrangement problem considered here by independently conquering the same problem on the smaller instance whose induced bipartite graph is a connected component of $\mathcal{G}(\alpha, I)$. In the following, we assume that the induced $\mathcal{G}(\alpha, I)$ of a given instance α and I has exactly one connected component for simplifying the discussion of our algorithm. By Lemma 5, we have the following corollary immediately.

Corollary 1 *Given an instance α and I with $\mathcal{G}(\alpha, I)$ being connected, there is an optimal series of events for transforming α into I with $n_{fu} = \chi(\alpha) - 1$ and $n_{fi} = \chi(I) - 1$.*

According to the assumption (i.e., $\mathcal{G}(\alpha, I)$ is connected) and discussion above, there exists an optimal series $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_\delta \rangle$ of events which begins with $n_{fu} = \chi(\alpha) - 1$ fusions for joining all chromosomes of α into a single chromosome α', and then follows n_{bi} block-interchanges for mutating α' into I', and finally ends with $n_{fi} = \chi(I) - 1$ fissions for splitting I' into I. Moreover, σ can be expressed as a product of 2-cycles, say $c_\lambda c_{\lambda-1} \ldots c_1$, such that $I\alpha^{-1} = c_\lambda c_{\lambda-1} \ldots c_1$. By Lemma 1, $\lambda \ge n - f(I\alpha^{-1})$. Actually, we can show later that λ indeed equals to $n - f(I\alpha^{-1})$. By Corollary 1 as well as the fact that a block-interchange can be expressed by two consecutive 2-cycles, we have $n_{bi} = \frac{n - f(I\alpha^{-1}) - n_{fu} - n_{fi}}{2}$ and as a result, $\delta = \frac{n - f(I\alpha^{-1}) + n_{fu} + n_{fi}}{2}$.

Next, we shall find n_{fu} 2-cycles from $I\alpha^{-1}$ such that these 2-cycles can be served as the n_{fu} fusions to join all chromosomes of α into a single chromosome, if α has multiple chromosomes (i.e., $\chi(\alpha) > 1$). Given any cycle β, we use $x \in \beta$ to denote that x is a number in β. For any two $x \in \beta$ and $y \in \beta$, they are said to be *adjacent* in β if $\beta(x) = y$ or $\beta(y) = x$.

Lemma 6 *Let α and I be two permutations with $\mathcal{G}(\alpha, I)$ being connected, and let α_i and α_j be any two initial cycles in α. Then there must exist a cycle β in $I\alpha^{-1}$ that contains two numbers x and y such that x is in α_i and y is in α_j.*

Proof. Since the induced bipartite graph $\mathcal{G}(\alpha, I)$ contains exactly one connected component, α_i and α_j contain numbers u and v, respectively, such that both u and v are in a cycle I_k of I. Then we claim that there is a cycle β in $I\alpha^{-1}$ that contains two numbers x and y such that $x \in \alpha_i$ and $y \in \alpha_j$. Suppose that there is no cycle in $I\alpha^{-1}$ that contains x and y such that $x \in \alpha_i$ and $y \in \alpha_j$. Then for any cycle γ in $I\alpha^{-1}$, all numbers in γ are contained in some cycle of α. It implies that u and v are in the different cycles in $I\alpha^{-1}$ (otherwise, they will be in the same cycle in α, contradicting to $u \in \alpha_i$ and $v \in \alpha_j$). Let u be in the cycle $\gamma_1 = (u \equiv a_1, a_2, \ldots, a_p)$ of $I\alpha^{-1}$ and v be in another cycle $\gamma_2 = (v \equiv b_1, b_2, \ldots, b_q)$ of $I\alpha^{-1}$ such that $I\alpha^{-1}(a_p) = u$ and $I\alpha^{-1}(b_q) = v$. As discussed above, all numbers in γ_1 (respectively, γ_2) are contained in α_i (respectively, α_j). Then we have $u, a_p \in \alpha_i$ and $v, b_q \in \alpha_j$, and hence $b_q \notin \alpha_i$ and $a_p \notin \alpha_j$. Recall that u and v both are in I_k. If α_i does not contain any number in γ_2, then it is impossible that u and v both are in I_k, since in this case, only those numbers that are also in α_i have the opportunity of being in the cycle of I in which u is. Let α_i contains $b_r \in \gamma_2$. Then α_i also has to contain all numbers in γ_2. As a result, $b_q \in \alpha_i$, a contradiction. Hence, there exists a cycle β in $I\alpha^{-1}$ that contains x and y such that $x \in \alpha_i$ and $y \in \alpha_j$. $\square$

The following lemma can be easily verified.

Lemma 7 $(a_1, a_2, \ldots, a_i, \ldots, a_j) = (a_1, a_2, \ldots, a_i)(a_{i+1}, \ldots, a_j)(a_i, a_j)$, *where* $1 \leq i < j$.

According to Lemma 6, for any two cycles α_i and α_j of α, we can find two numbers x and y in a cycle β of $I\alpha^{-1}$ such that $x \in \alpha_i$ and $y \in \alpha_i$. Let $\beta = (b_1, b_2, \dots, b_j)$, where $j \geq 2$. Then we consider the following two cases. Case 1: x and y are adjacent in β. For simplicity, let $x = b_{j-1}$ and $y = b_j$. Then by Lemma 7, $\beta = (b_1, b_2, \dots, b_{j-1})(b_j)(x, y)$. Case 2: x and y are not adjacent in β. Without loss of generality, let $x = b_i$ and $y = b_j$, where $1 \leq i < b_{j-1}$. Then $\beta = (b_1, b_2, \dots, b_i)(b_{i+1}, \dots, b_j)(x, y)$ according to Lemma 7. In other words, we can derive a 2-cycle (x, y) from β such that it is able to join α_i and α_j into one cycle. After (x, y) operates on α_i and α_j, the number of the cycles (including 1-cycles) in the resulting $I\alpha^{-1}$ increases by one. Repeatedly based on the discussion above, we can derive consecutive n_{fu} 2-cycles from $I\alpha^{-1}$, say $\phi_1, \phi_2, \dots, \phi_{n_{fu}}$, that can join $\chi(\alpha)$ cycles in α into a single cycle, where $n_{fu} = \chi(\alpha) - 1$. In other words, $\phi_1, \phi_2, \dots, \phi_{n_{fu}}$ are n_{fu} fusions that are able to transform genome α with $\chi(\alpha)$ chromosomes into a genome, denoted by α', with a single chromosome. In this case, we have $\alpha' = \phi_{n_{fu}}\phi_{n_{fu}-1}\dots\phi_1\alpha$, $I\alpha'^{-1} = I\alpha^{-1}\phi_1\phi_2\dots\phi_{n_{fu}}$, and $f(I\alpha'^{-1}) = f(I\alpha^{-1}) + n_{fu}$. Hence, we have the following claim immediately.

Claim 1 *$\alpha' = \phi_{n_{fu}}\phi_{n_{fu}-1}\dots\phi_1\alpha$, $I\alpha'^{-1} = I\alpha^{-1}\phi_1\phi_2\dots\phi_{n_{fu}}$, and $f(I\alpha'^{-1}) = f(I\alpha^{-1}) + n_{fu}$, where $n_{fu} = \chi(\alpha) - 1$.*

Now, suppose that $\chi(I) > 1$. Then it is clear that the induced $\mathcal{G}(I, \alpha')$ is a connected bipartite graph. As the similar discussion before, we can derive consecutive n_{fi} 2-cycles from $\alpha' I^{-1}$, say $\psi_1, \psi_2, \dots, \psi_{n_{fi}}$, such that they serve as the fusions to transform I with $\chi(I)$ chromosomes into a genome, denoted by I', with only one chromosome, where $n_{fi} = \chi(I) - 1$ and $\alpha' I^{-1}$ is the inverse of $I\alpha'^{-1}$ (i.e., $\alpha' I^{-1} = (I\alpha'^{-1})^{-1}$). In addition, we have $I' = \psi_{n_{fi}}\psi_{n_{fi}-1}\dots\psi_1 I$ (hence $\psi_1\psi_2\dots\psi_{n_{fi}} I' = I$), $\alpha' I'^{-1} = \alpha' I^{-1}\psi_1\psi_2\dots\psi_{n_{fi}}$ and $f(\alpha' I'^{-1}) = f(\alpha' I^{-1}) + n_{fi}$. Conversely, it means that $\psi_{n_{fi}}, \psi_{n_{fi}-1}, \psi_1$ can be used to split I' with one chromosome into I with n_{fi} chromosomes. Since $\alpha' I^{-1}$ is the inverse of $I\alpha'^{-1}$, it can be easily obtained from $I\alpha'^{-1}$ by just reversing the order of the numbers in each cycle of $I\alpha'^{-1}$, and hence $f(\alpha' I^{-1}) = f(I\alpha'^{-1})$, which leads to $f(\alpha' I'^{-1}) = f(I\alpha'^{-1}) + n_{fi}$. As a result, we have $f(I'\alpha'^{-1}) = f(\alpha' I^{-1}) = f(I\alpha'^{-1}) + n_{fi}$, since $I'\alpha'^{-1} = (\alpha' I^{-1})^{-1}$. In summary, the following claim is immediate.

Claim 2 *$\psi_1\psi_2\ldots\psi_{n_{fi}}I' = I$, $\alpha' I'^{-1} = \alpha' I^{-1}\psi_1\psi_2\ldots\psi_{n_{fi}}$ and $f(I'\alpha'^{-1}) = f(I\alpha'^{-1}) + n_{fi}$, where $n_{fi} = \chi(I) - 1$.*

Notice that both α' and I' now are a genome with exactly one chromosome. Then based on the algorithm proposed by Lin *et al.* [Lin *et al.*, 2005], we can find $n_{bi} = \frac{n-f(I'\alpha'^{-1})}{2}$ block-interchanges from $I'\alpha'^{-1}$ to transform α' into I'. Certainly, these n_{bi} block-interchanges can be further expressed by a product of $2n_{bi}$ 2-cycles, say $\tau_1^1, \tau_1^2, \tau_2^1, \tau_2^2, \ldots, \tau_{n_{bi}}^1, \tau_{n_{bi}}^2$, such that every two consecutive 2-cycles act as a block-interchange in the process of transforming α' into I' and $I'\alpha'^{-1} = \tau_{n_{bi}}^2\tau_{n_{bi}}^1\tau_{n_{bi}-1}^2\tau_{n_{bi}-1}^1\ldots\tau_1^2\tau_1^2$. Hence, we have the following claim immediately.

Claim 3 *$I' = \tau_{n_{bi}}^2\tau_{n_{bi}}^1\tau_{n_{bi}-1}^2\tau_{n_{bi}-1}^1\ldots\tau_1^2\tau_1^1\alpha'$.*

Let $\rho = \psi_1\psi_2\ldots\psi_{n_{fi}}\tau_{n_{bi}}^2\tau_{n_{bi}}^1\ldots\tau_1^2\tau_1^1\phi_{n_{fu}}\phi_{n_{fu}-1}\ldots\phi_1$. Then the result of $\rho\alpha = I$ (hence $\rho = I\alpha^{-1}$) can be easily verified by Claims 1, 2 and 3 as follows.

$$\begin{aligned}
\rho\alpha &= \psi_1\psi_2\ldots\psi_{n_{fi}}\tau_{n_{bi}}^2\tau_{n_{bi}}^1\ldots\tau_1^2\tau_1^1\phi_{n_{fu}}\phi_{n_{fu}-1}\ldots\phi_1\alpha \\
&= \psi_1\psi_2\ldots\psi_{n_{fi}}\tau_{n_{bi}}^2\tau_{n_{bi}}^1\ldots\tau_1^2\tau_1^1\alpha' && \text{(by Claim 1)} \\
&= \psi_1\psi_2\ldots\psi_{n_{fi}}I' && \text{(by Claim 3)} \\
&= I && \text{(by Claim 2)}
\end{aligned}$$

In other words, ρ is a feasible series of rearrangement events that can transform α into I, and can be expressed by a product of $(n_{fu} + (n - f(I'\alpha'^{-1})) + n_{fi})$ 2-cycles. More clearly, ρ first uses $\phi_1, \phi_2, \ldots, \phi_{n_{fu}}$ (acting as n_{fu} fusions) to transform α into α', then uses $\tau_1^1, \tau_1^2, \ldots, \tau_{n_{bi}}^1, \tau_{n_{bi}}^2$ (acting as n_{bi} block-interchanges) to transform α' into I', and finally uses $\psi_{n_{fi}}, \psi_{n_{fi}-1}, \ldots, \psi_1$ (acting as n_{fi} fissions) to transform I' into I. By Claims 1 and 2, we can show that $n_{fu} + (n - f(I'\alpha'^{-1})) + n_{fi} = n - f(I\alpha^{-1})$ as follows.

$$\begin{aligned}
& n_{fu} + (n - f(I'\alpha'^{-1})) + n_{fi} \\
=\ & n_{fu} + (n - f(I\alpha'^{-1}) - n_{fi}) + n_{fi} && \text{(by Claim 2)} \\
=\ & n_{fu} + (n - f(I\alpha^{-1}) - n_{fu} - n_{fi}) + n_{fi} && \text{(by Claim 1)} \\
=\ & n - f(I\alpha^{-1})
\end{aligned}$$

According to Lemma 1, $n - f(I\alpha^{-1})$ is the minimum number such that $I\alpha^{-1}$ can be expressed as a product of $n - f(I\alpha^{-1})$ 2-cycles. By Corollary 1, we can conclude that ρ is an optimal series of events that can transform α into I with n_{fu} fusions, $\frac{n-f(I\alpha^{-1})-n_{fu}-n_{fi}}{2}$ block-interchanges and n_{fi} fissions, where $n_{fu} = \chi(\alpha) - 1$ and $n_{fi} = \chi(I) - 1$. Hence, the following lemma is immediate.

Lemma 8 *Let α and I be two multichromosomal genomes with $\mathcal{G}(\alpha, I)$ being connected. Then there is an optimal series of events needed to transform α into I such that all n_{fu} fusions come before all $\frac{n-f(I\alpha^{-1})-n_{fu}-n_{fi}}{2}$ block-interchanges that come before all n_{fi} fissions, where $n_{fu} = \chi(\alpha) - 1$ and $n_{fi} = \chi(I) - 1$.*

Let us take $\alpha = (1, 2, 10)(11, 8, 9, 3, 6)(7, 4, 5, 12)$ and $I = (1, 2, 3)(4, 5)(6, 7, 8)(9, 10, 11, 12)$ for an example. It is not hard to see that $\mathcal{G}(\alpha, I)$ is a connected bipartite graph with $\chi(\alpha) = 3$ and $\chi(I) = 4$, and $I\alpha^{-1} = (1, 11, 7, 9, 6)(3, 10)(4, 8, 12)$ and hence $f(I\alpha') = 5$, since two 1-cycles (i.e., (2) and (5)) are not explicitly shown. First, we are going to find $(n_{fu} = \chi(\alpha) - 1 = 2)$ 2-cycles ϕ_1 and ϕ_2 from $I\alpha^{-1}$ to transform α with three cycles into α' with exactly one cycle. Since $I\alpha^{-1} = (1, 11, 7, 9, 6)(4, 12)(4, 8)(3, 10)$, we let $\phi_1 = (3, 10)$ and $\phi_2 = (4, 8)$ and hence by Claim 1, $\alpha' = \phi_2\phi_1\alpha = (4, 5, 12, 7, 8, 9, 10, 1, 2, 3, 6, 11)$ and $I\alpha'^{-1} = I\alpha^{-1}\phi_1\phi_2 = (1, 11, 7, 9, 6)(4, 12)$. Next, we need to find $(n_{fi} = \chi(I) - 1 = 3)$ 2-cycles ψ_1, ψ_2 and ψ_3 from $\alpha' I^{-1}$, which is equal to $(I\alpha'^{-1})^{-1} = (6, 9, 7, 11, 1)(12, 4) = (1, 7, 11)(1, 9)(1, 6)(12, 4)$, to transform I into I' with only one cycle. By letting $\psi_1 = (12, 4), \psi_2 = (1, 6)$ and $\psi_3 = (1, 9)$, we have $I' = \psi_3\psi_2\psi_1 I = (1, 2, 3, 6, 7, 8, 9, 10, 11, 4, 5, 12)$ and $\alpha' I'^{-1} = \alpha' I^{-1}\psi_1\psi_2\psi_3 = (1, 7, 11)$ according to Claim 2. Finally, we will find $(n - f(I\alpha^{-1}) - n_{fu} - n_{fi} = 12 - 5 - 2 - 3 = 2)$ 2-cycles τ_1^1 and τ_1^2 from $I'\alpha'^{-1}$ that act as a block-interchange to transform α' into I', where $I'\alpha'^{-1} = (\alpha' I'^{-1})^{-1} = (11, 7, 1) = (11, 1)(11, 7)$. By letting $\tau_1^1 = (11, 7)$ and $\tau_1^2 = (11, 1)$, we have $\tau_1^2\tau_1^1\alpha' = (11, 1)(11, 7)(4, 5, 12, 7, 8, 9, 10, 1, 2, 3, 6, 11) = (11, 4, 5, 12, 1, 2, 3, 6, 7, 8, 9, 10)$, which indeed equals to I'. Consequently, we can find an optimal series of events $\rho = \psi_1\psi_2\psi_3\tau_1^2\tau_1^1\phi_2\phi 1$ that can transform α into I (i.e., $\rho\alpha = I$).

Based on the idea above, we design Algorithm Sorting-by-ffbi (meaning sorting by fusions, fissions and block-interchanges) to compute the genome rearrangement distance $d(\alpha, I)$ between two given multichromosomal genomes α and I and also generate an optimal scenario of the required rearrangement events. The purpose of step 2.3.3 is to find two numbers x and y that are both in some cycle of $\beta = I_i{\alpha_i}^{-1}$, but are in different cycles in α_i. By Lemma 6, such x and y exist only if $\mathcal{G}(\alpha_i, I_i)$ is connected. Actually, they can be found using the following simple approach. For simplicity, let $\beta_k = (b_k^1, b_k^2, \ldots, b_k^{l_k})$ be a cycle in β that contains two numbers x and y such that they are in different cycles in α_i. Then we only need to check if b_k^1 and b_k^j, where $2 \le j \le l_k$, are in different cycles in α_i or not. If so, we let $x = b_k^1$ and $y = b_k^j$. The reason is as follows. Suppose that both b_k^1 and b_k^j for all $2 \le j \le l_k$ are in the same cycle in α_i. Then all numbers $b_k^1, b_k^2, \ldots, b_k^{l_i}$ in β_k are in the same cycle in α_i, which contradicts to the assumption that there are x and y in β_k that are in different cycles in α_i.

```
**Algorithm Sorting-by-ffbi**
**Input:** Two multichromosomal genomes $\alpha$ and $I$.
**Output:** $d(\alpha, I)$ and a minimum series $\sigma$ of operations required to transform $\alpha$ into $I$.
1:      Find all connected components $P_1, P_2, \ldots, P_\omega$ in graph $\mathcal{G}(\alpha, I)$;
        Let $n_i = |\mathsf{gene}(P_i, \alpha)|$ for each $1 \le i \le \omega$;
2:      for each $P_i$, $1 \le i \le \omega$, do
            /* Let $\alpha_i$ (resp. $I_i$) be the collection of chromosomes in $\alpha$ (resp. $I$) whose
            corresponding vertices are in $P_i$. */
2.1:        Compute $I_i{\alpha_i}^{-1}$ and let $\beta = I_i{\alpha_i}^{-1}$;
2.2:        $n_{fu} = \chi(\alpha_i) - 1$, $n_{fi} = \chi(I_i) - 1$, $n_{bi} = \frac{n_i - f(\beta) - n_{fu} - n_{fi}}{2}$ and $\delta_i = n_{fu} + n_{bi} + n_{fi}$;
2.3:        if $\chi(\alpha_i) > 1$ then   /* To compute $\phi_1, \phi_2, \ldots, \phi_{n_{fu}}$ */
2.3.1:          for each cycle of $\alpha_i$ do
                    Create a set to contain all the numbers in this cycle;
                endfor
2.3.2:          $k = 1$ and $h = 2$;
2.3.3:          for $j = 1$ to $n_{fu}$ do
                    /* Assume $\beta = \beta_1\beta_2 \ldots \beta_p$ and $\beta_k = (b_k^1, b_k^2, \ldots, b_k^{l_k})$ with each $l_k \ge 2$ */
                    $S = \mathsf{find\text{-}set}(b_k^1)$;
                    while $(S = \mathsf{find\text{-}set}(b_k^h))$ do
                        if $h < l_k$ then $h = h + 1$; else $k = k + 1$, $h = 2$ and $S = \mathsf{find\text{-}set}(b_k^1)$;
                    endwhile
                    $x = b_k^1$ and $y = b_k^h$;
                    $\phi_j = (x, y)$ and $\mathsf{union}(x, y)$;
                endfor
2.3.4:          $\alpha'_i = \phi_{n_{fu}}\phi_{n_{fu}-1} \ldots \phi_1\alpha_i$ and $\beta = \beta\phi_1\phi_2 \ldots \phi_{n_{fu}}$;   /* Currently, $\beta$ is $I{\alpha'_i}^{-1}$ */
            endif
```

```
2.4:        if $\chi(I_i) > 1$ then    /* To compute $\psi_1, \psi_2, \ldots, \psi_{n_{fi}}$ */
2.4.1:          $\beta = \beta^{-1}$;    /* New $\beta$ becomes $\alpha'_i I^{-1}$ */
2.4.2:          for each cycle of $I_i$ do
                    Create a set to contain all the numbers in this cycle;
                endfor
2.4.3:          $k = 1$ and $h = 2$;
2.4.4:          for $j = 1$ to $n_{fi}$ do
                    /* Assume $\beta = \beta_1\beta_2 \ldots \beta_q$ and $\beta_k = (b_k^1, b_k^2, \ldots, b_k^{l_k})$ with each $l_k \geq 2$ */
                    $S = \mathsf{find\text{-}set}(b_k^1)$;
                    while ($S = \mathsf{find\text{-}set}(b_k^h)$) do
                        if $h < l_k$ then $h = h + 1$; else $k = k + 1$, $h = 2$ and $S = \mathsf{find\text{-}set}(b_k^1)$;
                    endwhile
                    $x = b_k^1$ and $y = b_k^h$;
                    $\psi_j = (x, y)$ and $\mathsf{union}(x, y)$;
                endfor
2.4.5:          $I'_i = \psi_{n_{fi}}\psi_{n_{fi}-1} \ldots \psi_1 I_i$ and $\beta = \beta\psi_1\psi_2 \ldots \psi_{n_{fi}}$;    /* Currently, $\beta$ is $\alpha'_i I_i'^{-1}$ */
            endif
2.5:        /* To compute $\tau_1^1, \tau_i^2, \ldots, \tau_{n_{bi}}^1, \tau_{n_{bi}}^2$ */
2.5.1:      $\beta = \beta^{-1}$;    /* New $\beta$ becomes $I_i{}'\alpha_i'^{-1}$ */
2.5.2:      $n_{bi} = \frac{n_i - f(\beta)}{2}$;
2.5.3:      for $j = 1$ to $n_{bi}$ do
                Arbitrarily choose two adjacent elements $x$ and $y$ in $\beta$;
                /* Let $\alpha' = (a_1, a_2, \ldots, a_{n_i})$. */
                Circularly shift $(a_1, a_2, \ldots, a_{n_i})$ such that $a_1 = x$ and assume $y = a_k$;
                $\tau_j^1 = (x, y)$;
                for $h = 1$ to $n_i$ do
                    $\mathsf{index}(a_h) = h$;
                end for
                Find two adjacent elements $u$ and $v$ in $\beta(x, y)$ such that
                $\mathsf{index}(u) \leq k - 1$ and $\mathsf{index}(v) \geq k$;
                $\tau_j^2 = (u, v)$;
                $\alpha' = \tau_j^2\tau_j^1\alpha'$ and $\beta = \beta\tau_j^1\tau_j^2$;
            endfor
3:      Let $\sigma_i = \psi_1 \ldots \psi_{n_{fi}}\tau_{n_{bi}}^2\tau_{n_{bi}}^1 \ldots \tau_1^2\tau_1^1\phi_{n_{fu}} \ldots \phi_1$;
4:      Output $d(\alpha, I) = \sum_{i=1}^{\omega} \delta_i$ and $\sigma = \sigma_1\sigma_2 \ldots \sigma_\omega$;
```

Theorem 1 *Given two circular multichromosomal genomes α and I over the same gene set $E = \{1, 2, \ldots, n\}$, the problem of computing the genome rearrangement distance between α and I using fusions, fissions and block-interchanges and an optimal series of such operations can be solved in $\mathcal{O}(n^2)$ time.*

Proof. As discussed previously, Algorithm Sorting-by-ffbi transforms α into I using a minimum of fusion/fission/block-interchange operations. Below, we analyze its time-complexity. Notice that given an undirected graph $\mathcal{H}$ with p vertices and q edges, the connected components in $\mathcal{H}$ can be found in $\mathcal{O}(p+q)$ time using depth-first search or breadth-first search. Hence, step 1 can be done in $\mathcal{O}(n^2)$ time for computing the connected components in the induced bipartite graph $\mathcal{G}(\alpha, I)$, since in the worst case, the number of edges in $\mathcal{G}$ is $\chi(\alpha) \times \chi(I)$, and $\chi(\alpha) = \mathcal{O}(n)$ and $\chi(I) = \mathcal{O}(n)$. In step 2, there are ω outer iterations, each with computing the a minimum series of events needed to transform α_i into I_i, where $1 \leq i \leq \omega$. Clearly, steps 2.1 costs $\mathcal{O}(n_i)$ time for computing $I_i \alpha_i^{-1}$ and step 2.2 takes only a constant time. The cost of step 2.3 is dominated by that of step 2.3.3. There are n_{fu} inner iterations in step 2.3, each with the purpose of finding two numbers x and y that are both in the same cycle in β, but in the different cycles in α_i. In the worst case, step 2.3 needs n_i find-set operations and n_{fu} union operations to finish its overall process. Note that step 2.3.1 can be implemented by initially creating a set for each number in $\mathsf{gene}(P_i, \alpha)$ and then performing $n_i - \chi(\alpha_i)$ union operations to generate $\chi(\alpha_i)$ sets with each corresponding to a cycle in α_i, where $\chi(\alpha_i) = n_{fu} + 1$. Hence, the total number of union operations is $n_i - 1$ in step 2.3. Actually, these find-set and union operations can be implemented in $\mathcal{O}(n_i)$ time using the so-called static disjoint set union and find algorithm proposed by Gabow and Tarjan [Gabow & Tarjan, 1985]. In other words, step 2.3 costs only $\mathcal{O}(n_i)$ time. Similarly, it can be verified that the cost of step 2.4 is $\mathcal{O}(n_i)$ time. As to step 2.5, it is adopted from the paper of Lin *et al.* [Lin *et al.*, 2005] and takes $\mathcal{O}(n_{bi} n_i)$ time, where $n_{bi} = \frac{n_i - f(I_i{}' \alpha_i'^{-1})}{2} = \mathcal{O}(n_i)$. As a result, the cost of step 2 is $\mathcal{O}(\zeta n)$, where ζ is the maximum n_{bi} among all iterations in step 2 and $\zeta < n$. Clearly, steps 3 and 4 cost a constant time. Therefore, the total time-complexity of Algorithm $\mathsf{Sorting\text{-}by\text{-}ffbi}$ is $\mathcal{O}(n^2)$ time. $\square$

Chapter 5

Discussion and Conclusion

In this thesis, we studied the block-interchange distance problem form the algebraic view and developed a web server called ROBIN, which automatically search for the identical homologous/conserved regions shared by all the input sequences. For testing the applicability of ROBIN, we apply ROBIN to genomes of three human *Vibrio* pathogens for detecting the evolutionary relationships. Consequently, our experimental results coincide with the previous ones obtained by other community using a comparative genomic approach, which implies that the block-interchange events seem to play a significant role in the evolution of *Vibrio* species. We also studied the genome rearrangement problem for two circular multichromosomal genomes by simultaneously considering fusion, fission and block-interchange events. We showed that there is an optimal series of events required to transform one genome into another such that all fusions come before all block-interchanges which come before all fissions. Based on this property as well as the concept of permutation groups in algebra, we designed an $\mathcal{O}(n^2)$ time algorithm for efficiently computing the minimum number of fusions, fissions and block-interchanges needed to transform one circular multichromosomal genome into another and also generating an optimal scenario of the required rearrangement events. In addition, we have implemented this algorithm as a web server (This web server is freely available at http://genome.life.nctu.edu.tw/FFBI). With these two tools, biologists can conduct the analysis of genome rearrangements either intra-chromosomal by block-interchanges or inter-chromosomal by fusions, fissions and block-interchanges to their biological data of interest.

References

[Bader *et al.*, 2001] Bader, D. A., Yan, M. & Moret, B. M. W. (2001) A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology,* **8**, 483–491.

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

[Bafna & Pevzner, 1998] Bafna, V. & Pevzner, P. A. (1998) Sorting by transpositions. *SIAM Journal on Discrete Mathematics,* **11**, 221–240.

[Berman & Hannenhalli] Berman, P. & Hannenhalli, S. (1996) Fast sorting by reversal. In *Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching (CPM1996)*, (Hirschberg, D. S. & Myers, E., eds), vol. 1075, of *Lecture Notes in Computer Science* pp. 168–185 Springer-Verlag.

In *Proceedings of the 7th Annual Symposium on Combinatorial Pattern Matching (CPM1996)*, (Hirschberg, D. S. & Myers, E., eds), vol. 1075, of *Lecture Notes in Computer Science* pp. 168–185 Springer-Verlag.

[Berman *et al.*] Berman, P., Hannenhalli, S. & Karpinski, M. (2002) 1.375-approximation algorithm for sorting by reversals. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA2002)*, (Mohring, R. H. & Raman, R., eds), vol. 2461, of *Lecture Notes in Computer Science* pp. 200–210 Springer-Verlag.

[Brudno *et al.*, 2003] Brudno, M., Do, C. B., Cooper, G. M., Kim, M. F., Davydov, E., Green, E. D., Sidow, A. & Batzoglou, S. (2003) LAGAN and Multi-LAGAN: efficient tools for large-scale multiple alignment of genomic DNA. *Genome Research,* **13**, 721–731.

[Caprara, 1997] Caprara, A. (1997) Sorting by reversal is difficult. In *Proceedings of the 1th Annual International Conference on Research in Computational Molecular Biology (RECOMB1997)* pp. 75–83 ACM Press.

[Caprara, 1999] Caprara, A. (1999) Sorting permutations by reversals and eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics,* **12**, 91–110.

[Chen *et al.*, 2003] Chen, C. Y., Wu, K. M., Chang, Y. C. & Chang, C. H. (2003) Comparative genome analysis of *Vibrio vulnificus*, a marine pathogen. *Genome Research,* **13**, 2577–2587.

[Christie, 1996] Christie, D. A. (1996) Sorting by block-interchanges. *Information Processing Letters,* **60**, 165–169.

[Christie, 1998] Christie, D. A. (1998) A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA1998)* pp. 244–252 ACM/SIAM.

[Darling *et al.*, 2004*a*] Darling, A. C. E., Mau, B., Blattner, F. R. & Perna, N. T. (2004*a*) Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome Research,* **14**, 1394–1403.

[Darling *et al.*, 2004*b*] Darling, A. E., Mau, B., Blattner, F. R. & Perna, N. T. (2004*b*) GRIL: genome rearrangement and inversion locator. *Bioinformatics,* **20**, 122–124.

[Delcher *et al.*, 1999] Delcher, A. L., Kasif, S., Fleischmann, R. D., Peterson, J., White, O. & Salzberg, S. L. (1999) Alignment of whole genomes. *Nucleic Acids Research,* **27**, 2369–2376.

[Delcher *et al.*, 2002] Delcher, A. L., Phillippy, A., Carlton, J. & Salzberg, S. L. (2002) Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research,* **30**, 2478–2483.

[Fraleigh, 1999] Fraleigh, J. B. (1999) *A First Course in Abstruct Algebra.* 6th edition,, Addison-Wesley.

[Gabow & Tarjan, 1985] Gabow, H. N. & Tarjan, R. E. (1985) A linear-time algorithm for a special case of disjoint set union. *Journal of Computer and System Sciences,* **30**, 209–221.

[Hannenhalli, 1996] Hannenhalli, S. (1996) Polynomial algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics,* **71**, 137–151.

[Hannenhalli & Pevzner, 1995] Hannenhalli, S. & Pevzner, P. A. (1995) Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science (FOCS1995)* pp. 581–592 IEEE Computer Society.

[Hannenhalli & Pevzner, 1999] Hannenhalli, S. & Pevzner, P. A. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM,* **46**, 1–27.

[Kaplan *et al.*, 2000] Kaplan, H., Shamir, R. & Tarjan, R. E. (2000) Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing,* **29**, 880–892.

[Kececioglu & Ravi, 1995] Kececioglu, J. D. & Ravi, R. (1995) Of mice and men: algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms (SODA1995)* pp. 604–613 ACM/SIAM, San Francisco.

[Kececioglu & Sankoff, 1993] Kececioglu, J. D. & Sankoff, D. (1993) Exact and approximation algorithms for the inversion distance between two permutations. In *Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching (CPM1993),* (Apostolico, A., Crochemore, M., Galil, Z. & Manber, U., eds), vol. 684, of *Lecture Notes on Computer Science* pp. 87–105 Springer-Verlag.

[Lin *et al.*, 2005] Lin, Y. C., Lu, C. L., Chang, H.-Y. & Tang, C. Y. (2005) An efficient algorithm for sorting by block-interchanges and its application to the evolution of vibrio species. *Journal of Computational Biology,* **12**, 102–112.

[Lu *et al.*, 2005] Lu, C. L., Wang, T. C., Lin, Y. C., & Tang, C. Y. ROBIN: a tool for genome rearrangement of block-interchanges. *Bioinformatics*, In press.

[Meidanis & Dias, 2000] Meidanis, J. & Dias, Z. (2000) An alternative algebraic formalism for genome rearrangements. In *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families*, (Sankoff, D. & Nadeau, J. H., eds), pp. 213–223 Kluwer Academic Publisher.

[Meidanis & Dias, 2001] Meidanis, J. & Dias, Z. (2001) Genome rearrangements distance by fusion, fission, and transposition is easy. In *Proceedings of the 8th International Symposium on String Processing and Information Retrieval (SPIRE2001)*, (Navarro, G., ed.), pp. 250–253 IEEE Computer Society.

[Morgenstern, 1999] Morgenstern, B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics,* **15**, 211–218.

[Morgenstern *et al.*, 1998] Morgenstern, B., Frech, K., Dress, A. & Werner, T. (1998) DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics,* **14**, 290–294.

[Pevzner & Tesler, 2003] Pevzner, P. & Tesler, G. (2003) Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome Research,* **13**, 37–45.

[Schwartz *et al.*, 2003] Schwartz, S., Kent, W. J., Smit, A., Zhang, Z., Baertsch, R., Hardison, R. C., Haussler, D. & Miller, W. (2003) Human-mouse alignments with BLASTZ. *Genome Research,* **13**, 103–107.

[Tesler, 2002] Tesler, G. (2002) GRIMM: genome rearrangements web server. *Bioinformatics,* **18**, 492–493.

[Walter *et al.*, 1998] Walter, M. E. M. T., Dias, Z. & Meidanis, J. (1998) Reversal and transposition distance of linear chromosomes. In *Proceedings of String Processing and Information Retrieval (SPIRE1998)* pp. 96–102 IEEE Computer Society.