JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 1, 2005

Pp. 102–112

An Efficient Algorithm for Sorting by Block-Interchanges and Its Application to the Evolution of Vibrio Species

YING CHIH LIN,[1] CHIN LUNG LU,[2] HWAN-YOU CHANG,[3] and CHUAN YI TANG[1]

ABSTRACT

In the study of genome rearrangement, the block-interchanges have been proposed recently as a new kind of global rearrangement events affecting a genome by swapping two nonintersecting segments of any length. The so-called block-interchange distance problem, which is equivalent to the sorting by block-interchange problem, is to find a minimum series of block-interchanges for transforming one chromosome into another. In this paper, we study this problem by considering the circular chromosomes and propose a $\mathcal{O}(\delta n)$ time algorithm for solving it by making use of permutation groups in algebra, where n is the length of the circular chromosome and δ is the minimum number of block-interchanges required for the transformation, which can be calculated in $\mathcal{O}(n)$ time in advance. Moreover, we obtain analogous results by extending our algorithm to linear chromosomes. Finally, we have implemented our algorithm and applied it to the circular genomic sequences of three human vibrio pathogens for predicting their evolutionary relationships. Consequently, our experimental results coincide with the previous ones obtained by others using a different comparative genomics approach, which implies that the block-interchange events seem to play a significant role in the evolution of vibrio species.

Key words: genome rearrangement, sorting by block-interchanges, sorting by transpositions, permutation group, vibrio genomes.

1. INTRODUCTION

With large amounts of various genomic data (DNA, RNA, and protein sequences) becoming available, the study of genome rearrangement, which is the measurement of the evolutionary difference between two organisms by conducting large scale comparisons of their genomic data, has been drawing a lot of attentions in computational biology. One of the most promising ways to do this research is to compare the orders of the identical genes in two different genomes. Unlike from the traditional point

[1]Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C.
[2]Department of Biological Science and Technology, National Chiao Tung University, Hsinchu 300, Taiwan, R.O.C.
[3]Department of Life Science, National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C.

mutations such as insertions, deletions, and substitutions, various large scale mutations, such as reversals (Bader *et al.*, 2001; Bafna and Pevzner, 1996; Berman and Hannenhalli, 1996; Berman and Karpinski, 1999; Berman *et al.*, 2002; Caprara, 1997, 1999; Christie, 1998; El-Mabrouk, 2000; Hannenhalli and Pevzner, 1999; Kaplan *et al.*, 2000; Kececioglu and Sankoff, 1993; Siepel, 2002), transpositions (Bafna and Pevzner, 1998; Christie, 1999; Eriksen, 2002; Gu *et al.*, 1999; Hartman, 2003; Lin and Xue, 2001; Meidanis *et al.*, 1997, 2002; Walter *et al.*, 1998; 2000), translocations (Hannenhalli, 1996; Kececioglu and Ravi, 1995), fissions and fusions (Hannenhalli and Pevzner, 1995; Meidanis and Dias, 2001, 2002) and block-interchanges (Christie, 1996), acting on genes within or among chromosomes, have been proposed to determine the evolutionary distance between two related genomes by comparing the gene orders.

It is well known that the reversal distance problem is equal to the *sorting by reversal problem*, which is to find a minimum number of reversals for transforming one permutation into another one. For two unsigned permutations, Caprara (1997) first showed this problem to be NP-hard, and Berman and Karpinski (1999) later proved it to be MAX-SNP hard. On the other hand, Kececloglu and Sankoff (1993) gave a 2-approximation algorithm, which was further improved to a factor of 1.75 by Bafna and Pevzner (1996), then to a factor of 1.5 by Christie (1998), and finally to 1.375 by Berman *et al.* (2002). As to the same problem with two signed permutations, however, Hannenhalli and Pevzner (1999) first presented a polynomial-time algorithm, whose time-complexity is $\mathcal{O}(n^4)$, which was subsequently improved to $\mathcal{O}(n^2\alpha(n))$ by Berman and Hannenhalli (1996) and to $\mathcal{O}(n^2)$ by Kaplan *et al.* (2000) and Bader *et al.* (2001), where n is the length of the permutation and α is the inverse Ackerman function.

Similarly, the so-called *sorting by transposition problem* is to determine a minimum number of transpositions for converting one permutation into another. Whether or not this problem between two unsigned permutations is NP-complete is still open. However, Bafna and Pevzner (1998), Christie (1999), and Hartman (2003) gave several approximation algorithms for this problem, in which the best one has the performance ratio of 1.5 with time-complexity $\mathcal{O}(n^2)$. Using the breakpoint diagram approach, Walter *et al.* (2000) presented a simpler approximation algorithm with a performance ratio of 2.25 and time-complexity of $\mathcal{O}(b^2)$, where b is the number of breakpoints in the diagram.

As to the *translocation distance problem*, which is to find the minimum translocation distance among multiple chromosomes, Kececioglu and Ravi (1995) first gave a 2-approximation algorithm, if the orientations of genes in the chromosomes are unknown. If the orientations of genes are known, Hannenhalli's (1996) duality theorem leads to a polynomial-time algorithm for solving the problem.

The *sorting by block-interchange problem*, proposed by Christie (1996), is to compute a minimum number of block-interchanges for transforming one permutation into another one. By modeling two linear chromosome as two unsigned permutations, Christie (1996) proposed an $\mathcal{O}(n^2)$ time algorithm to exactly solve the problem using the breakpoint diagram approach. In this paper, we study this problem by considering the permutations as circular chromosomes. By making use of the permutation groups in algebra, we design a very simple and efficient algorithm for solving it 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. We also extend our algorithm to deal with the case of linear chromosomes and obtain analogous results. It is worth mentioning that any algorithm for optimally solving the sorting by block-interchange problem can serve as a 2-approximation algorithm for the sorting by transposition problem. The reason is that the block-interchange exchanging two nonintersecting segments of any length is a general case of the transposition with which the exchanged segments must be adjacent, and any block-interchange can be replaced with at most two transpositions. In addition, we have implemented our algorithm of sorting by block-interchange as a computer program and applied it to the circular genomic sequences of three human vibrio pathogens, including *V. vulnificus*, *V. parahaemolyticus*, and *V. cholerae* (see Section 4 for details), for predicting their evolutionary relationships. Consequently, our experimental results show that 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*, which indeed meet with the previous results obtained by Chen *et al.* (2003) using a different comparative genomics approach. This coincidence seems to indicate that the block-interchange events play a significant role in the evolution of vibrio species.

The rest of this paper is organized as follows. In Section 2, we introduce some basic concepts about permutation groups in algebra and describe its relationship with genome rearrangement. In Section 3, we first present our algorithm to solve the sorting by block-interchange problem for circular chromosomes and then

describe its extensions to deal with the case of linear chromosomes. The application of our developed program for the case of circular chromosomes will be presented in Section 4. Finally, we make a conclusion in Section 5.

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 instance, we may define a permutation α of the set $\{1, 2, 3, 4, 5, 6\}$ by specifying $\alpha(1) = 4, \alpha(2) = 3, \alpha(3) = 1$, $\alpha(4) = 2, \alpha(5) = 6$, and $\alpha(6) = 5$. This correspondence is usually expressed in the following *array form*, where $\alpha(i)$ is placed directly below i for each $i \in E$.

$$\alpha = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 3 & 1 & 2 & 6 & 5 \end{bmatrix}$$

More practically, the above permutation α is expressed using a so-called *cycle notation* as illustrated in Fig. 1 and simply denoted by $\alpha = (1, 4, 2, 3)(5, 6)$. In the literature, the form $(a_1, a_2, \ldots, a_k)$ is used to denote a *cycle of length k* (or *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 $(a_k, a_1, a_2, \ldots, a_{k-1})$. As illustrated above, it is not hard to see that a permutation is composed of one or more cycles and can be expressed by a product of cycles.

Any two cycles are said to be *disjoint* if they have no element in common. Note that any permutation can be written in a unique way as the product of disjoint cycles (if we ignore the order of the cycles in the product) (Fraleigh, 1999; Meidanis and Dias, 2000). This product of disjoint cycles is also called the *cycle decomposition* of the permutation. 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 $\alpha(x) = x$. In particular, 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 $x \in E$. For example, let $E = \{1, 2, 3, 4, 5, 6\}$, $\alpha = (2, 3, 4)$, and $\beta = (3, 1, 5, 2, 6, 4)$. Then we have $\alpha\beta = (1, 5, 3)(2, 6)$. Note that we have $\alpha\beta = \beta\alpha$ only when α and β are disjoint cycles. The *inverse* of a permutation α is defined to be a permutation, denoted by α^{-1}, such that $\alpha\alpha^{-1} = \alpha^{-1}\alpha = \mathbf{1}$. Note that $\alpha^{-1} = \alpha$ if α is a 2-cycle. In fact, every permutation has a unique inverse (Fraleigh, 1999). If a permutation is expressed by the product of disjoint cycles, then its inverse can be obtained by simply reverting the order of the elements in each cycle. For instance, if $\alpha = (1, 4, 2, 3)(5, 6)$, then $\alpha^{-1} = (3, 2, 4, 1)(6, 5)$.

Meidanis and Dias (2000, 2001) first noticed that the permutation groups play a very important role in the study of genome rearrangement. They observed 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 also found that the global evolutionary events, such as fusions and fissions (respectively, transpositions), correspond to the composition of a 2-cycle (respectively, 3-cycle) and the permutation corresponding to a genome. For example, given a permutation α whose cycle decomposition is $c_1c_2\cdots c_r$:

- If $\rho = (x, y)$ is a 2-cycle and x and y are in the same cycle, 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 $(y \equiv a_i, a_{i+1}, \ldots, a_j)$. For instance, if $\rho = (1, 3)$ and $\alpha = (1, 4, 5, 3, 2)$, then we have $\rho\alpha = (1, 4, 5)(3, 2)$. In this case, ρ is a fission event for α, and for simplicity, we call ρ a *split operation* of α (or c_p).
- If $\rho = (x, y)$ is a 2-cycle and x and y are in 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 $(x \equiv a_1, a_2, \ldots, a_i, y \equiv b_1, b_2, \ldots, b_j)$. For instance, if $\rho = (1, 2)$ and $\alpha = (1, 4, 5)(3, 2)$, then we have $\rho\alpha = (1, 4, 5, 2, 3)$. In this case, ρ is a fusion event for α, and we call ρ a *join operation* of α (or c_p and c_q).
- If $\rho = (x, y, z)$ is a 3-cycle and x, y, and z are in the same cycle, 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 $(x \equiv a_1, a_2, \ldots, a_i, z \equiv c_1, c_2, \ldots, c_k, y \equiv b_1, b_2, \ldots, b_j)$. For instance, if $\rho = (1, 3, 5)$ and $\alpha = (1, 2, 3, 4, 5, 6)$, then we have $\rho\alpha = (1, 2, 5, 6, 3, 4)$. In this case, ρ is a transposition event for α.

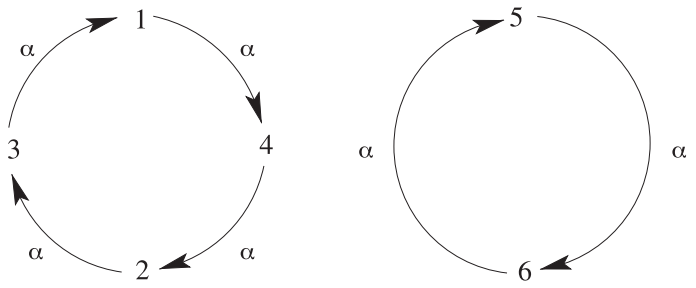

FIG. 1. The illustration of a permutation $\alpha = (1, 4, 2, 3)(5, 6)$ meaning that $\alpha(1) = 4, \alpha(2) = 3, \alpha(3) = 1$, $\alpha(4) = 2, \alpha(5) = 6$, and $\alpha(6) = 5$.

The *block-interchange* event introduced by Christie (1996) is a generalization of a transposition event and affects a chromosome by swapping two nonintersecting segments of any length, where the swapped segments are not necessarily adjacent in the chromosome. In this paper, we observe that the block-interchange event for a circular chromosome is able to be modeled by two consecutive 2-cycles affecting its corresponding permutation as follows. Let ρ_1 and ρ_2 be two 2-cycles and α be a permutation of a circular chromosome. If ρ_1 is a split operation of α and ρ_2 is a join operation of $\rho_1\alpha$, then the composition $\rho_2\rho_1\alpha$ is a result of a block-interchange event affecting α. For instance, if $\rho_1 = (1, 4)$, $\rho_2 = (5, 6)$, and $\alpha = (1, 5, 2, 4, 7, 6, 3)$, then $\rho_2\rho_1\alpha = (1, 6, 3, 4, 7, 5, 2)$, which is equal to the permutation obtained from α by exchanging two nonintersecting blocks [6, 3] and [5, 2]. Now, we explain the block-interchanged result of $\rho_2\rho_1\alpha$ in the following. Let $\alpha = (a_1, a_2, \ldots, a_k)$. Recall that we assume that ρ_1 is a split operation of α and ρ_2 is a join operation of $\rho_1\alpha$. Without loss of generality, we let $\rho_1 = (a_1, a_x)$ and $\rho_2 = (a_y, a_z)$, where $1 < x \leq k$ and $1 \leq y < z \leq k$. After ρ_1 affects α, we obtain two disjoint cycles $(a_1, a_2, \ldots, a_{x-1})$ and $(a_x, a_{x+1}, \ldots, a_k)$ (i.e., $\rho_1\alpha = (a_1, a_2, \ldots, a_{x-1})(a_x, a_{x+1}, \ldots, a_k)$). Since ρ_2 is a join operation of $\rho_1\alpha$, y and z must be in the different cycles in $\rho_1\alpha$, and hence we have $1 \leq y \leq x - 1$ and $x \leq z \leq k$. For simplicity, we rewrite this as $\rho_1\alpha = (a_y, \ldots, a_{x-1}, a_1, \ldots, a_{y-1})(a_z, \ldots, a_k, a_x, \ldots, a_{z-1})$. Then we have $\rho_2\rho_1\alpha = (a_y, \ldots, a_{x-1}, a_1, \ldots, a_{y-1}, a_z, \ldots, a_k, a_x, \ldots, a_{z-1}) = (a_1, \ldots, a_{y-1}, \boxed{a_z, \ldots, a_k}, a_x, \ldots, a_{z-1}, \boxed{a_y, \ldots, a_{x-1}})$. In other words, we can obtain $\rho_2\rho_1\alpha$ from α by exchanging the blocks $[a_y, a_{x-1}]$ and $[a_z, a_k]$. Hence, $\rho_2\rho_1$ is a block-interchange event for α and we have the following lemma immediately.

Lemma 1. *Let $\alpha = (a_1, a_2, \ldots, a_k)$, $\rho_1 = (a_1, a_x)$, and $\rho_2 = (a_y, a_z)$, where $1 < x \leq k$, $1 \leq y \leq x - 1$, and $x \leq z \leq k$. Then $\rho_2\rho_1\alpha$ can be obtained from α by exchanging the blocks $[a_y, a_{x-1}]$ and $[a_z, a_k]$.*

Conversely, given a block-interchange event σ affecting α, we can find two 2-cycles ρ_1 and ρ_2 such that ρ_1 is a split operation of α, ρ_2 is a join operation of $\rho_1\alpha$, and $\rho_2\rho_1\alpha$ is the result obtained from α by the block-interchange event σ. For convenience, we use $\sigma \otimes \alpha$ to denote a block-interchange event σ affecting a permutation α.

Lemma 2. *Let α be the permutation corresponding to a circular chromosome. For any arbitrary block-interchange event σ affecting α, we can find two 2-cycles ρ_1 and ρ_2 such that ρ_1 is a split operation of α, ρ_2 is a join operation of $\rho_1\alpha$, and $\sigma \otimes \alpha = \rho_2\rho_1\alpha$.*

3. SORTING A PERMUTATION BY BLOCK-INTERCHANGES

Given two permutations $\alpha = (a_1, a_2, \ldots, a_n)$ and $\beta = (b_1, b_2, \ldots, b_n)$ of $E = \{1, 2, \ldots, n\}$, the *block-interchange distance problem* is to find a minimum series of block-interchanges $\sigma_1, \sigma_2, \ldots, \sigma_t$ such that $\sigma_t \otimes \sigma_{t-1} \otimes \ldots \otimes \sigma_1 \otimes \alpha = \beta$ (i.e., transforming α into β), and the number t is called the *block-interchange distance* between α and β. Usually, β is replaced with $I = (1, 2, \ldots, n)$, and then the block-interchange distance problem can be considered as a problem of sorting a permutation using the minimum block-

interchanges. In this section, we will describe an efficient algorithm to find a minimum series of block-interchanges $\sigma_1, \sigma_2, \ldots, \sigma_t$ such that $\sigma_t \otimes \sigma_{t-1} \otimes \ldots \otimes \sigma_1 \otimes \alpha = I$. The fact $I\alpha^{-1}\alpha = I$ implies intuitively that $I\alpha^{-1}$ contains all information needed to transform α into I. Indeed, from $I\alpha^{-1}$, we are able to derive a minimum of block-interchanges to transform α into I.

Suppose that $\{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ is a series of block-interchanges that transforms α into I. By Lemma 2, for each σ_i, where $1 \leq i \leq k$, we are able to find two corresponding 2-cycles ρ_1^i and ρ_2^i such that $\sigma_i \otimes \sigma_{i-1} \otimes \ldots \otimes \sigma_1 \otimes \alpha = \rho_2^i \rho_1^i \rho_2^{i-1} \rho_1^{i-1} \ldots \rho_2^1 \rho_1^1 \alpha$. Then we have $\rho_2^k \rho_1^k \rho_2^{k-1} \rho_1^{k-1} \ldots \rho_2^1 \rho_1^1 = I\alpha^{-1}$, which means that $I\alpha^{-1}$ can be expressed by a product of an even number of 2-cycles. It is well known in abstract algebra that any permutation of a finite set of at least two elements is a product of 2-cycles and such an expression is not unique. However, if $\alpha_1\alpha_2 \ldots \alpha_x$ and $\beta_1\beta_2 \ldots \beta_y$ are two different products of 2-cycles of a permutation, then x and y are both even or both odd (Fraleigh, 1999). In other words, if we express $I\alpha^{-1}$ as a product of 2-cycles, then the number of 2-cycles must be even.

Lemma 3. *The number of 2-cycles in any product of 2-cycles of $I\alpha^{-1}$ is even.*

Given a k-cycle $\beta = (b_1, b_2, \ldots, b_k)$ with $k \geq 2$, it is not hard to verify that $\beta = (b_1, b_k)(b_1, b_{k-1}) \ldots (b_1, b_2)$. Then we call such a product of 2-cycles a *simple 2-cycle expression* of β and denote it by $\mathsf{sim}_2(\beta)$. Suppose that $C_1C_2 \ldots C_p$ is the cycle decomposition of $I\alpha^{-1}$ with each $l(C_i) \geq 2$, where $l(C_i)$ denotes the cycle length of C_i for each $1 \leq i \leq p$. Note that all cycles of length one are not expressed explicitly in the above cycle decomposition. Let $\lambda = \sum_{1 \leq i \leq p}(l(C_i) - 1)$. For convenience, we also let $\mathsf{sim}_2(I\alpha^{-1}) = \mathsf{sim}_2(C_1)\mathsf{sim}_2(C_2) \ldots \mathsf{sim}_2(C_p)$, and clearly $\mathsf{sim}_2(I\alpha^{-1})$ is a product of λ 2-cycles and $I\alpha^{-1} = \mathsf{sim}_2(I\alpha^{-1})$.

Let $f(I\alpha^{-1})$ denote the number of the disjoint cycles in the cycle decomposition of $I\alpha^{-1}$. Note that $f(I\alpha^{-1})$ counts also the nonexpressed cycles of length one. For example, if $\beta = (1, 5)(2, 4)$ is a permutation of $E = \{1, 2, \ldots, 5\}$, then $f(\beta) = 3$, instead of $f(\beta) = 2$, since $\beta = (1, 5)(2, 4)(3)$.

Lemma 4. *Let $\mathsf{sim}_2(I\alpha^{-1}) = \alpha_\lambda\alpha_{\lambda-1} \ldots \alpha_1$, where α_i is a 2-cycle for each $1 \leq i \leq \lambda$. Then λ is even and $\lambda + f(I\alpha^{-1}) = n$.*

Proof. Let $C_1C_2 \ldots C_p$ be the cycle decomposition of $I\alpha^{-1}$ with q fixed elements $x_1, x_2, \ldots, x_q$ not written explicitly (i.e., $f(I\alpha^{-1}) = p + q$). By definition, we have $\lambda = \sum_{1 \leq i \leq p}(l(C_i) - 1)$. In fact, each fixed element x_i in $I\alpha^{-1}$ can be expressed as a product of two 2-cycles such as $(x, y)(x, y)$, where $1 \leq i \leq q$ and y is an arbitrary element in E. In other words, $\alpha_\lambda\alpha_{\lambda-1} \ldots \alpha_1(x_1, y)(x_1, y) \ldots (x_q, y)(x_q, y)$ is a product of 2-cycles of $I\alpha^{-1}$, and hence λ is even according to Lemma 3. Since any two cycles of $C_1, C_2, \ldots, C_p$ are mutually disjoint, $n - q = \sum_{1 \leq i \leq p} l(C_i)$. As a result, we have $\lambda + f(I\alpha^{-1}) = n$. ■

Lemma 5. *If $\beta_p\beta_{p-1} \ldots \beta_1$ is an arbitrary product of 2-cycles of $I\alpha^{-1}$, then we have $p \geq \lambda$.*

Proof. Let $\mathsf{sim}_2(I\alpha^{-1}) = \alpha_\lambda\alpha_{\lambda-1} \ldots \alpha_1$. Recall that $\mathsf{sim}_2(I\alpha^{-1}) = I\alpha^{-1}$. Hence, we have $\mathbf{1} = (\alpha_\lambda\alpha_{\lambda-1} \ldots \alpha_1)^{-1} I\alpha^{-1} = \alpha_1^{-1}\alpha_2^{-1} \ldots \alpha_\lambda^{-1} I\alpha^{-1} = \alpha_1\alpha_2 \ldots \alpha_\lambda I\alpha^{-1}$, where $\mathbf{1} = (1)(2) \ldots (n)$. That is, applying $\alpha_1\alpha_2 \ldots \alpha_\lambda$ to $I\alpha^{-1}$ leads to all elements in E to be fixed. Similarly, all elements in E will be fixed if we apply $\beta_1\beta_2 \ldots \beta_p$ to $I\alpha^{-1}$ since $\mathbf{1} = \beta_1\beta_2 \ldots \beta_p I\alpha^{-1}$. By the constructive property of $\mathsf{sim}_2(I\alpha^{-1})$, each α_i, $1 \leq i \leq \lambda$, is always a split operation of some cycle of length greater than one in $\alpha_{i+1}\alpha_{i+2} \ldots \alpha_\lambda I\alpha^{-1}$. However, each β_i, $1 \leq i \leq p$, may be either a split operation or a join operation to $\beta_{i+1}\beta_{i+2} \ldots \beta_p I\alpha^{-1}$. It clearly implies that $p \geq \lambda$. ■

According to Lemma 2, any series of block-interchanges transforming α into I can be expressed by a product of 2-cycles of $I\alpha^{-1}$. By Lemmas 4 and 5, $\mathsf{sim}_2(I\alpha^{-1})$ is a product of 2-cycles of $I\alpha^{-1}$ with the minimum number of 2-cycles, where the number of 2-cycles in $\mathsf{sim}_2(I\alpha^{-1})$ is λ. For convenience, we let $\delta = \frac{\lambda}{2}$, since λ is even by Lemma 4. In fact, we can show later that there exists a product of 2-cycles of $I\alpha^{-1}$, say $\rho_2^\delta\rho_1^\delta \ldots \rho_2^1\rho_1^1$, such that for each $1 \leq i \leq \delta$, $\rho_2^i\rho_1^i$ is a block-interchange event for $\rho_2^{i-1}\rho_1^{i-1} \ldots \rho_2^1\rho_1^1\alpha$. In other words, $\rho_2^\delta\rho_1^\delta \ldots \rho_2^1\rho_2^1\alpha = I$, which means that we can transform α into I

using δ block-interchanges. Hence, we can conclude that δ is the block-interchange distance between α and I and $\delta = \frac{n-f(I\alpha^{-1})}{2}$ according to Lemma 4. Then we have the following theorem.

Theorem 1. *The block-interchange distance between α and I is $\delta = \frac{n-f(I\alpha^{-1})}{2}$.*

Next, we describe the method of finding a product of 2-cycles of $I\alpha^{-1}$, say $\rho_2^\delta\rho_1^\delta \dots \rho_2^1\rho_1^1$, such that for each $1 \le i \le \delta$, σ_i is a block-interchange event for $\sigma_{i-1} \otimes \sigma_{i-2} \otimes \dots \otimes \sigma_1 \otimes \alpha$, where $\sigma_i = \rho_2^i\rho_1^i$. First, we show the existence of such block-interchanges $\sigma_\delta, \sigma_{\delta-1}, \dots, \sigma_1$ for α in the following. Given any arbitrary permutation β, we say x and y are *adjacent* in β if $\beta(x) = y$ or $\beta(y) = x$. Particularly, if $\beta(x) = y$, then we further say that x *immediately precedes* y and y *immediately succeeds* x and denote such a relationship by $x \xrightarrow{\beta} y$ for the sake of clearness. Moreover, we use $\mathsf{n2c}(\beta)$ to denote the number of 2-cycles in $\mathsf{sim}_2(\beta)$.

Suppose that y immediately precedes x in α (i.e., $y \xrightarrow{\alpha} x$). Then we have $\alpha(y) = x$ and hence $\alpha^{-1}(x) = y$ and $I\alpha^{-1}(x) = z$, where $z = y + 1$ if $1 \le y < n$; otherwise, $z = 1$, which means that x immediately precedes z, instead of y, in $I\alpha^{-1}$. In other words, there are no two adjacent elements x and y in $I\alpha^{-1}$ with $x \xrightarrow{I\alpha^{-1}} y$ which are also adjacent in α such that y immediately precedes x. Let $C_1C_2 \dots C_p$ be the cycle decomposition of $I\alpha^{-1}$, where each C_i, $1 \le i \le p$, has length of greater than one. Then $\lambda = \mathsf{n2c}(I\alpha^{-1}) = 2\delta$. For simplicity, we let $n_i = l(C_i)$ for each $1 \le i \le p$ and let $C_i = (c_1^i, c_2^i, \dots, c_{n_i}^i)$. Without loss of generality, let $x = c_1^p$ and $y = c_2^p$. Then $x \xrightarrow{I\alpha^{-1}} y$ and $C_p = C'_p(x, y)$, where $C'_p = (x, c_3^p, \dots, c_{n_p}^p)$ if $n_p \ge 3$; otherwise, $C_p = \emptyset$. Clearly, (x, y) is a split operation to α since α is a single cycle containing x and y. For convenience, we further let $D_i = C_i$ for $1 \le i \le p-1$, $D_p = C'_p$ and $\alpha = (a_1 \equiv x, a_2, \dots, a_n)$. As discussed above, y is not equal to a_n, and hence we let $y = a_k$, where $2 \le k \le n-1$. Then $(x, y)\alpha = \alpha_1\alpha_2$, where $\alpha_1 = (x, a_2, \dots a_{k-1})$ and $\alpha_2 = (y, a_{k+1}, \dots, a_n)$. As a result, we have $I = I\alpha^{-1}\alpha = D_1D_2 \dots D_p(x, y)\alpha = D_1D_2 \dots D_p\alpha_1\alpha_2$, where clearly $D_1, D_2, \dots D_p$ are mutually disjoint and α_1 and α_2 are disjoint. Let $\beta = D_1D_2 \dots D_p$. It is worth mentioning that y is fixed in β (i.e., it does not belong to any cycle of $D_1, D_2, \dots, D_p$) and x is also fixed in β if $D_p = \emptyset$. Next, we claim that there exists at least a pair of two elements u and v with $\beta(u) = v$ (i.e., u immediately precedes v in some cycle of $D_1D_2 \dots D_p$) such that (u, v) is a join operation to $\alpha_1\alpha_2$. Suppose that there exists no such a pair of two elements u and v. Then for any two elements c and d with $\beta(c) = d$, both c and d either belong to α_1 or belong to α_2. Recall that α_1 and α_2 are disjoint and $\alpha_1 = (a_1, a_2, \dots, a_{k-1})$, where $k \le n-1$. For simplicity of illustration, we assume that $a_i < a_{i+1} < n$ for $1 \le i \le k-2$, which leads to the conclusion that $a_{k-1} + 1$ is in α_2. Then we have $\alpha_1(a_i) = a_{i+1}$ and $\beta(a_{i+1}) = a_i + 1$ for $1 \le i \le k-2$ and $\alpha_1(a_{k-1}) = a_1$ and $\beta(a_1) = a_{k-1} + 1$. As a result, all elements in $\{a_i + 1 : 1 \le i \le k-1\}$ belong to α_1, which contradicts the fact that $a_{k-1} + 1$ is in α_2. In other words, there is at least a pair of two adjacent elements u and v in β such that u belongs to α_1 and v belongs to α_2 (i.e., (u, v) is a join operation to $\alpha_1\alpha_2$). Then we have $I = \beta\alpha_1\alpha_2 = \beta(u, v)^{-1}(u, v)\alpha_1\alpha_2 = \beta(u, v)(u, v)\alpha_1\alpha_2$, where $(u, v)\alpha_1\alpha_2$ forms a single cycle with n elements, and both u and v will be fixed in $\beta(u, v)$ if some cycle $D_q = (u, v)$, $1 \le q \le p$; otherwise, only v will be fixed by $\beta(u, v)$. It is not hard to see that $\mathsf{n2c}(I\alpha^{-1}) = \mathsf{n2c}(\beta(u, v)) + 2$. As discussed above, we have derived two 2-cycles ρ_1^1 and ρ_2^1 from $I\alpha^{-1}$ such that $\sigma_1 = \rho_2^1\rho_1^1$ is a block-interchange operation to α, where $\rho_1^1 = (x, y)$ and $\rho_2^1 = (u, v)$. Moreover, we can reformulate $I = I\alpha^{-1}\alpha$ into $I = I\gamma^{-1}\gamma$ such that $\mathsf{n2c}(I\alpha^{-1}) = \mathsf{n2c}(I\gamma^{-1}) + 2$, where $\gamma = \rho_2^1\rho_1^1\alpha$ and $I\gamma^{-1} = \beta\rho_2^1$. By continuing in the way discussed above, we are able to finally find δ block-interchange operations $\sigma_1, \sigma_2, \dots, \sigma_\delta$ to transform α into I.

Based on the discussion above, we are able to use δ block-interchange operations, say, $\sigma_1, \sigma_2, \dots, \sigma_\delta$, for optimally transforming α into I. Moreover, for each $1 \le i \le \delta$, σ_i can be derived from $\beta = I\alpha^{-1}\rho_1^{-1}\rho_2^{-1} \dots \rho_{i-1}^{-1}$ by first choosing any two adjacent elements x and y in β and letting $\rho_1^i = (x, y)$, and then finding any two adjacent elements u and v in $\beta(x, y)$ such that (u, v) is a join operation of $(x, y)(\sigma_{i-1}\otimes\sigma_{i-2}\otimes\dots\otimes\sigma_1\otimes\alpha)$ and letting $\rho_2^i = (u, v)$. Let us take $\alpha = (4, 2, 1, 3, 6, 5, 8, 7)$ for an example. Then we have $I\alpha^{-1} = (1, 3, 2, 5, 7)(4, 8, 6)$. By Theorem 1, we understand that the block-interchange distance between α and I is $\delta = \frac{8-2}{2} = 3$, which means that α can be transformed into I using three block-interchange operations. Next, we show how to find these three block-interchange operations $\sigma_1 =$

$\rho_2^1\rho_1^1$, $\sigma_2 = \rho_2^2\rho_1^2$, and $\sigma_3 = \rho_2^3\rho_1^3$. Initially, we choose $\rho_1^1 = (1, 3)$ arbitrarily so that we obtain $I\alpha^{-1} = (1, 2, 5, 7)(1, 3)(4, 8, 6)$ and then $I\alpha^{-1}\alpha = (1, 2, 5, 7)(4, 8, 6)(1)(3, 6, 5, 8, 7, 4, 2)$. Next, we find two adjacent elements, say, $(1, 2)$, in $(1, 2, 5, 7)(4, 8, 6)$ because $(1, 2)$ is a join operation of $(1)(3, 6, 5, 8, 7, 4, 2)$ and let $\rho_2^1 = (1, 2)$. As a result, we have $I\alpha^{-1}\alpha = (1, 5, 7)(4, 8, 6)(1, 2)(1)\ (3, 6, 5, 8, 7, 4, 2) = (1, 5, 7)(4, 8, 6)(1, 2, 3, 6, 5, 8, 7, 4)$. At the second iteration, we let $\rho_1^2 = (1, 5)$ and $\rho_2^2 = (1, 7)$ and then obtain $I\alpha^{-1}\alpha = (1, 7)(1, 5)(4, 8, 6)\ (1, 2, 3, 6, 5, 8, 7, 4) = (4, 8, 6)(1, 2, 3, 6, 7, 4, 5, 8)$. Finally, by letting $\rho_1^3 = (4, 8)$ and $\rho_2^3 = (4, 6)$, we have $I\alpha^{-1}\alpha = (4, 6)(4, 8)(1, 2, 3, 6, 7, 4, 5, 8) = (1, 2, 3, 4, 5, 6, 7) = I$. The details of our method above for solving the block-interchange distance problem is described in Algorithm Sorting by Block-Interchange.

Algorithm Sorting by Block-Interchange
Input: A permutation $\alpha = (a_1, a_2, \ldots, a_n)$;
Output: A minimum series of block-interchange operations $\sigma_1, \sigma_2, \ldots, \sigma_\delta$ for transforming α into I.
1: Let $\delta = \frac{n - f(I\alpha^{-1})}{2}$;
2: **for** $i = 1$ to δ **do**
2.1: Arbitrarily choose two adjacent elements x and y in $I\alpha^{-1}$;
2.2: Circularly shift $(a_1, a_2, \ldots, a_n)$ such that $a_1 = x$ and assume $y = a_k$;
2.3: **for** $j = 1$ to n **do**
$\mathsf{index}(a_j) = j$;
end for
2.4: Find two adjacent elements u and v in $I\alpha^{-1}(x, y)$ such that $\mathsf{index}(u) \leq k - 1$ and $\mathsf{index}(v) \geq k$;
2.5: $\sigma_i = (u, v)(x, y)$;
2.6: Compute $(u, v)(x, y)\alpha$ and denote it by α again;
2.7: Compute $I\alpha^{-1}(u, v)(x, y)$ and denote it by $I\alpha^{-1}$ again;
end for
3: Output $\sigma_1, \sigma_2, \ldots, \sigma_\delta$;

Theorem 2. *The block-interchange distance problem for a circular chromosome can be solved by Algorithm Sorting by Block-Interchange in $\mathcal{O}(\delta n)$ time.*

Proof. As discussed previously, Algorithm Sorting by Block-Interchange transforms α into I using a minimum number of block-interchange operations. We analyze the time-complexity of Algorithm Sorting by Block-Interchange as follows. It is not hard to see that the computation of step 1 can done in $\mathcal{O}(n)$ time. As to step 2, there are δ iterations, and in each iteration, each of substeps 2.1 to 2.7 costs $\mathcal{O}(n)$ time. As a result, the time-complexity of step 2 is $\mathcal{O}(n)$. Hence, the total time-complexity of Algorithm Sorting by Block-Interchange is $\mathcal{O}(\delta n)$, where $\delta = \frac{n - f(I\alpha^{-1})}{2}$. ■

Next, we describe how to slightly modify Algorithm Sorting by Block-Interchange to deal with the case of representing the permutation $\alpha = (a_1, a_2, \ldots, a_n)$ as a linear chromosome, instead of a circular chromosome, of a genome. In the beginning, we add a new element 0 into the beginning of α and denote this new permutation by $\alpha' = (0 \equiv a_0, a_1, a_2, \ldots, a_n)$. Next, we consider α' as a circular chromosome and apply the modified Algorithm Sorting by Block-Interchange (which we will introduce later) to α' such that the minimum block-interchange operations, say, $\sigma_1', \sigma_2', \ldots, \sigma_\delta'$, for optimally transforming α' into $I' = \{0, 1, \ldots, n\}$ satisfy the property that none of the two blocks interchanged by each σ_i' contains a_0, where $1 \leq i \leq \delta$. The first purpose of this property is to make sure that for each σ_i' affecting $\sigma_{i-1}' \otimes \ldots \otimes \sigma_1' \otimes \alpha'$, we can find a corresponding block-interchange operation σ_i to affect $\sigma_{i-1} \otimes \ldots \otimes \sigma_1 \otimes \alpha$ such that the blocks interchanged by σ_i are the same as the ones interchanged by σ_i'. The second purpose of the property is to guarantee that after applying all $\sigma_1, \sigma_2, \ldots, \sigma_i$ to α, the resulting permutation is I, which is due to the property that a_0 is not involved in any block-interchange. Finally, we can conclude that $\sigma_1, \sigma_2, \ldots, \sigma_\delta$ is a minimum block-interchange operations for transforming α into I, since any series

of block-interchanges of transforming α into I is also a series of block-interchanges of transforming α' into I'.

In the following, we describe how to modify Algorithm Sorting by Block-Interchange so that none of the two blocks interchanged by each σ'_i contains a_0, where $1 \leq i \leq \delta$. When applying the original Algorithm Sorting by Block-Interchange to α', the chosen (x, y) and (u, v) for the i iteration of step 2 (i.e., $\sigma'_i = (u, v)(x, y)$) may lead one of two interchanged nonempty blocks, say B_1 and B_2, to contain a_0. If this situation occurs, then we consider the following two cases. Case 1: B_1 and B_2 are not adjacent from the circular viewpoint. Then we just interchange the roles of (x, y) and (u, v) by setting (u, v) as the split operation and setting (x, y) as the join operation and then apply $(x, y)(u, v)$, instead of $(u, v)(x, y)$, to $\sigma'_{i-1} \otimes \ldots \otimes \sigma'_1 \otimes \alpha'$. We illustrate the reason for this by simply considering the iteration of producing σ'_1 as follows. Let $(x, y) = (a_i, a_j)$ and $(u, v) = (a_k, a_l)$, where $1 \leq i < j \leq n$, $i < k < j$ and $j < l$. Then it is not hard to see that $(u, v)(x, y)\alpha' = (a_i, a_{i+1}, \ldots, a_{k-1}, \boxed{a_l, a_{l+1}, \ldots, a_n, a_0, a_1 \ldots, a_{i-1}}, a_j, a_{j+1}, \ldots, a_{l-1}, \boxed{a_k, a_{k+1}, \ldots, a_{j-1}})$, and $(x, y)(u, v)\alpha' = (a_k, a_{k+1}, \ldots, a_{j-1}, \boxed{a_i, a_{i+1}, \ldots, a_{k-1}}, a_l, a_{l+1}, \ldots, a_n, a_0, a_1, \ldots, a_{i-1}, \boxed{a_j, a_{j+1}, \ldots, a_{l-1}})$. From the viewpoint of the circular chromosome, we have $(u, v)(x, y)\alpha' = (x, y)(u, v)\alpha'$, which is consistent with the fact that (x, y) and (u, v) are disjoint cycles and hence $(u, v)(x, y) = (x, y)(u, v)$. As a result, none of the blocks interchanged by $(x, y)(u, v)$ contains a_0. Case 2: B_1 and B_2 are adjacent, where we let B_1 be the block of $a_l, a_{l+1}, \ldots, a_n, a_0, a_1, \ldots, a_{i-1}$ and B_3 denote the remaining block. In this case, we have either $x = u$ or $y = v$. For the former case, it is not hard to see that $(u, v)(x, y) = (x, v)(x, y) = (x, y)(v, y)$ and applying $(x, y)(v, y)$, instead of $(u, v)(x, y)$, to α' leads to the exchange of B_2 and B_3, which is equivalent to the exchange of B_1 and B_2 from the circular viewpoint. Moreover, neither B_2 nor B_3 contains a_0. For the latter case, we have $(u, v)(x, y) = (u, y)(x, y) = (u, x)(u, y)$, and applying $(u, x)(u, y)$ leads to the exchange of B_2 and B_3.

As discussed above, we are able to compute the block-interchange distance between linear chromosomes α and I, which is equal to the block-interchange distance δ between circular chromosomes α' and I' and can be calculated in $\mathcal{O}(\delta n)$ time, also including minimum block-interchange operations for transforming α into I. In other words, the algorithm for solving the sorting by block-interchange problem for circular chromosomes can be used to solve the same problem for linear chromosomes, and vice versa. Hence, we have the following theorem.

Theorem 3. *The sorting by block-interchange problem for linear chromosomes is equivalent to the sorting by block-interchange problem for circular chromosomes.*

4. EXPERIMENTAL RESULT

According to the algorithm we described in the previous section, we have implemented a computer program[1] for calculating the block-interchange distance between two circular or linear chromosomes with a series of the corresponding block-interchange operations for transforming one chromosome into another. Then we apply this program to predict the evolutionary relationships among three human vibrio pathogens, including *V. vulnificus*, *V. parahaemolyticus*, and *V. cholerae*. 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 (Chen *et al.*, 2003; Heidelberg *et al.*, 2000; Makino *et al.*, 2003) (see Table 1 for their sequence information). As more and more sequence information of vibrio species becomes available, a comparative genomics approach is needed 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.* (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

[1]This program is freely available at the website *http://algorithm.cs.nthu.edu.tw/tools/SORTBI/*.

Table 1. The Sequence Information of Three Pathogenic Vibrio Species, Each with Two Circular Chromosomes

Accession NO	*Species*	*Chromosome*	*Size (Mbps)*
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

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.* (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.* (2003).

The detailed steps we adopted in this experiment are as follow. First, we use a computer program called COSINE (*www.vein.cs.pu.edu.tw/*), a tool for finding consensuses or signatures in multiple sequences, to find the common fragments of sequences, each with fixed length of 17 bps, among the genomes of *V. vulnificus*, *V. parahaemolyticus*, and *V. cholerae*. As a result, for example, we obtained 13,259 (respectively, 233) such conserved fragments among the genomic sequences of VV1, VP1, and VC1 (respectively, VV2, VP2, and VC2). Among these conserved fragments, some fragments may appear more than once and/or overlap with each other in the genomic sequence. Then we remove those repeated fragments for the sake of simplicity and further merge those overlapped fragments into a new and larger one. In the end, there are 1,032 (respectively, 54) conserved fragments of length 17 to 140 (respectively, 17 to 26) bps remained for VV1, VP1, and VC1 (respectively, VV2, VP2, and CV2), Next, we apply our developed program to each instance for computing the block-interchange distances of each pair of vibrio species. Consequently, as shown in Table 2, the block-interchange events seem to occur frequently in genomes of vibrio species, and in both circular chromosomes, 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 other words, our experimental results indeed coincide with those obtained by Chen *et al.* (2003) using a different comparative genomics approach. This coincidence seems to indicate that the block-interchange events may play a significant role in the evolution of vibrio species. With more and more vibrio and other bacterial genomes being available, we will be able to conduct the same experiments on a larger scale. We believe that our developed algorithms and programs in this paper will benefit the biologist for studies of the evolution and even the biological functions of bacteria or higher organisms.

Table 2. The Block-Interchange Distances among VV1, VP1, and VC1 (left) and among VV2, VP2, and VC2 (right)

	VV1	VP1	VC1		VV2	VP2	VC2
VV1	—	39	69	VV2	—	3	6
VP1	39	—	65	VP2	3	—	7
VC1	69	65	—	VC2	6	7	—

5. CONCLUSIONS

In this paper, we studied the block-interchange distance problem for two circular chromosomes, which is equivalent to the problem of sorting a permutation α by block-interchange, from the algebraic viewpoint. We showed that the block-interchange distance of α is $\delta = \frac{n-f(I\alpha^{-1})}{2}$, which can be computed in $\mathcal{O}(n)$ time. Moreover, we proposed an $\mathcal{O}(\delta n)$ time algorithm for finding such a series of δ block-interchanges to transform α into I. We also extended our algorithm to solve the same problem for linear chromosomes and obtained analogous results. Finally, we implemented our algorithm of sorting by block-interchange and applied it to the circular genomic sequences of three human vibrio pathogens to show their evolutionary relationships based on the calculated block-interchange distances. Consequently, our experimental results coincide with previous results obtained by Chen *et al.* (2003) using a different comparative genomics approach, which seems to imply that the block-interchange events play a significant role in the evolution of vibrio species.

REFERENCES

Bader, D.A., Yan, M., and Moret, B.M.W. 2001. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comp. Biol.* 8, 483–491.

Bafna, V., and Pevzner, P.A. 1996. Genome rearrangements and sorting by reversals. *SIAM J. Comput.* 25, 272–289.

Bafna, V., and Pevzner, P.A. 1998. Sorting by transpositions. *SIAM J. Dis. Math.* 11, 221–240.

Berman, P., and Hannenhalli, S. 1996. Fast sorting by reversal. *Proc. 7th Ann. Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science,* 1075, 168–185.

Berman, P., Hannenhalli, S., and Karpinski, M. 2002. 1.375-approximation algorithm for sorting by reversals. *Proc. 10th Ann. Eur. Symp. on Algorithms, Lecture Notes in Computer Science,* 2461, 200–210.

Berman, P., and Karpinski, M. 1999. On some tighter inapproximability results. *Proc. 26th Int. Coll. on Automata, Languages and Programming, Lecture Notes in Computer Science,* 1644, 200–209.

Caprara, A. 1997. Sorting by reversal is difficult. *Proc. 1st Ann. Int. Conf. on Research in Computational Molecular Biology,* 75–83.

Caprara, A. 1999. Sorting permutations by reversals and eulerian cycle decompositions. *SIAM J. Dis. Math.* 12, 91–110.

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

Christie, D.A. 1996. Sorting by block-interchanges. *Inf. Process. Lett.* 60, 165–169.

Christie, D.A. 1998. A 3/2-approximation algorithm for sorting by reversals. *Proc. 9th Ann. ACM-SIAM Symp. on Discrete Algorithms,* ACM/SIAM, 244–252.

Christie, D.A. 1999. *Genome Rearrangement Problem.* Ph.D. Thesis, University of Glasgow.

El-Mabrouk, N. 2000. Genome rearrangement by reversals and insertions/deletions of contiguous segments. *Proc. 11th Ann. Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science,* 1848, 222–234.

Eriksen, N. 2002. $(1+\varepsilon)$-approximation of sorting by reversals and transpositions. *Theoret. Comput. Sci.* 289, 517–529.

Fraleigh, J.B. 1999. *A First Course in Abstruct Algebra,* 6th Ed., Addison-Wesley, Reading, MA.

Gu, Q.P., Peng, S., and Sudborough, H. 1999. A 2-approximation algorithms for genome rearrangements by reversals and transpositions. *Theoret. Comput. Sci.* 210, 327–339.

Hannenhalli, S. 1996. Polynomial algorithm for computing translocation distance between genomes. *Disc. Appl. Math.* 71, 137–151.

Hannenhalli, S., and Pevzner, P.A. 1995. Transforming men into mice (polynomial algorithm for genomic distance problem). *Proc. 36th IEEE Symp. on Foundations of Computer Science,* IEEE Computer Society, 581–592.

Hannenhalli, S., and Pevzner, P.A. 1999. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *J. ACM* 46, 1–27.

Hartman, T. 2003. A simpler 1.5-approximation algorithm for sorting by transposition. *Proc. 14th Ann. Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science,* 2676, 156–169.

Heidelberg, J.F., Eisen, J.A., Nelson, W.C., Clayton R.A., *et al.* 2000. DNA sequence of both chromosomes of the cholera pathogen *Vibrio cholerae*. *Nature* 406, 477–483.

Kaplan, H., Shamir, R., and Tarjan, R.E. 2000. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.* 29, 880–892.

Kececioglu, J.D., and Ravi, R. 1995. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms,* ACM/SIAM, 604–613.

Kececioglu, J.D., and Sankoff, D. 1993. Exact and approximation algorithms for the inversion distance between two permutations. *Proc. 4th Ann. Symp. on Combinatorial Pattern Matching, Lecture Notes in Computer Science*, 684, 87–105.

Lin, G.-H., and Xue, G. 2001. Signed genome rearrangement by reversals and transpositions: Models and approximations. *Theoret. Comput. Sci.* 259, 513–531.

Makino, K., Oshima, K., Kurokawa, K., Yokoyama, K., *et al.* 2003. Genome sequence of *Vibrio parahaemolyticus*: A pathogenic mechanism distinct from that of V cholerae. *Lancet* 361, 743–749.

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

Meidanis, J., and Dias, Z. 2001. Genome rearrangements distance by fusion, fission, and transposition is easy. *Proc. 8th Int. Symp. on String Processing and Information Retrieval*, IEEE Computer Society, 250–253.

Meidanis, J., and Dias, Z. 2002. The genome rearrangement distance by fusion, fission, and transposition with arbitrary weights. Technical report IC-02-01, Institute of Computing, University of Campinas.

Meidanis, J., Walter, M.E.M.T., and Dias, Z. 1997. Transposition distance between a permutation and its reverse. *Proc. 4th South American Workshop on String Processing*, Carleton University Press, 70–79.

Meidanis, J., Walter, M.E.M.T., and Dias, Z. 2002. A lower bound on the reversal and transposition diameter. *J. Comp. Biol.* 9, 743–746.

Siepel, A.C. 2002. An algorithm to find all sorting reversals. *Proc. 6th Ann. Int. Conf. on Computational Biology*, ACM Press, 281–290.

Walter, M.E.M.T., Dias, Z., and Meidanis, J. 1998. Reversal and transposition distance of linear chromosomes. *Proc. String Processing and Information Retrieval*, IEEE Computer Society, 96–102.

Walter, M.E.M.T., Dias, Z., and Meidanis, J. 2000. A new approach for approximating the transposition distance. *Proc. 7th Int. Symp. on String Processing and Information Retrieval*, IEEE Computer Society, 27–29.

Address correspondence to:
Chin Lung Lu
Department of Biological Science and Technology
National Chiao Tung University
Taiwan, R.O.C.

E-mail: cllu@mail.nctu.edu.tw

This article has been cited by:

1. Masud Hasan, Sohel RahmanAdvances in Genome Rearrangement Algorithms 749-772. [CrossRef]
2. Keng-Hsuan Huang, Kun-Tze Chen, Chin Lung Lu. 2011. Sorting permutations by cut-circularize-linearize-and-paste operations. *BMC Genomics* **12**:Suppl 3, S26. [CrossRef]
3. Y.-L. Huang, C.-C. Huang, C. Y. Tang, C. L. Lu. 2010. SoRT2: a tool for sorting genomes and reconstructing phylogenetic trees by reversals, generalized transpositions and translocations. *Nucleic Acids Research* **38**:Web Server, W221-W227. [CrossRef]
4. Yen-Lin Huang, Chin Lung Lu. 2010. Sorting by Reversals, Generalized Transpositions, and Translocations Using Permutation Groups. *Journal of Computational Biology* **17**:5, 685-705. [Abstract] [Full Text PDF] [Full Text PDF with Links]
5. Hao Zhao, Guillaume BourqueChromosomal Rearrangements in Evolution 165-182. [CrossRef]
6. Max A. Alekseyev. 2008. Multi-Break Rearrangements and Breakpoint Re-Uses: From Circular to Linear Genomes. *Journal of Computational Biology* **15**:8, 1117-1131. [Abstract] [Full Text PDF] [Full Text PDF with Links]
7. Matthieu Muffato, Hugues Roest Crollius. 2008. Paleogenomics in vertebrates, or the recovery of lost genomes from the mist of time. *BioEssays* **30**:2, 122-134. [CrossRef]
8. Michal Ozery-Flato, Ron Shamir. 2007. Sorting by Reciprocal Translocations via Reversals Theory. *Journal of Computational Biology* **14**:4, 408-422. [Abstract] [Full Text PDF] [Full Text PDF with Links]
9. Guillaume Bourque, Louxin ZhangModels and Methods in Comparative Genomics **68**, 59-104. [CrossRef]
10. Ying Chih Lin, Chuan Yi TangExposing Phylogenetic Relationships by Genome Rearrangement **68**, 1-57. [CrossRef]
11. S. Yancopoulos, O. Attie, R. Friedberg. 2005. Efficient sorting of genomic permutations by translocation, inversion and block interchange. *Bioinformatics* **21**:16, 3340-3346. [CrossRef]
12. C. L. Lu, T. C. Wang, Y. C. Lin, C. Y. Tang. 2005. ROBIN: a tool for genome rearrangement of block-interchanges. *Bioinformatics* **21**:11, 2780-2782. [CrossRef]