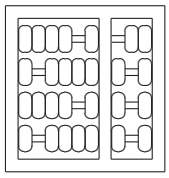


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Sorting by Fissions, Fusions, and Signed Reversals in $O(n)$

Cleber Mira *João Meidanis*

Technical Report - IC-14-12 - Relatório Técnico

August - 2014 - Agosto

Sorting by Fissions, Fusions, and Signed Reversals in $O(n)$

Cleber Mira* João Meidanis†

Abstract

Measuring the minimum number of rearrangement events that transforms a genome into another is an important tool for the comparative analysis of genomes. We propose a new algorithm for finding a minimum sequence of Fissions, Fusions, and Signed Reversals that transforms a genome into another. The algorithm is based on representing a chromosome as a cycle (a circular sequence) instead of a mapping. Thus a genome is a product of cycles instead of the usual representation as a set of mappings. By representing a chromosome as a cycle, rearrangement events like fissions, fusions, and signed reversals are performed in constant running time. Besides the change in the representation, the algorithm uses a hash table to deal with the problem of using the gene names for indexes. The total time complexity is $O(nh)$, where n is the number of genes and h is the time spent to retrieve data for a gene in the hash table (ideally h is a constant).

We also present a formula for the minimum number of Fissions, Fusions, and Signed Reversals that can be calculated in $O(nh)$ running time.

1 Introduction

The comparison of positions and orientations of genes in two distinct genomes may reveal relevant information about the genome's positions in a phylogeny, modifications of evolutionary hypothesis, and similarities among genome structures [17].

A genome of a species is a set of macromolecules (chromosomes) that encode in certain molecular segments (genes) the information necessary to produce every protein in a live being. We present two formal models for genes, chromosomes and genomes. The first one is based on the work of Hannenhalli and Pevzner [6] and we call it the *classical formalism*.

A *gene* is identified by an integer whose sign represents the orientation of the gene. A *chromosome* is a function that maps a *position* in the chromosome to the gene in that position; that is, for the chromosome π the gene found in the position i is π_i. The classical formalism assumes that a chromosome is *linear*, that is, there are two genes that are called the *extremities* of the chromosome. A *circular* chromosome does not have extremities, so in order to be represented by the classical formalism two adjacent genes are chosen to be

*Scylla Bioinformatics

†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

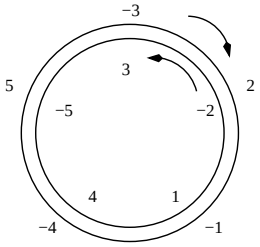
the extremities. A chromosome has two equivalent representations [6]: $\pi = [\pi_1, \ldots, \pi_n]$ and $-\pi = [-\pi_n, \ldots, -\pi_1]$. A chromosome π can also be represented by a permutation over $\{1, \ldots, 2n\}$, called the *image* of π, that consists of the sequence obtained from π by replacing π_i by $2\pi_i - 1$ and $2\pi_i$ when π_i is a positive element, and by $2\pi_i$ and $2\pi_i - 1$, otherwise. For instance, the image of $[-3, 2, -1, -4, -5]$ is the permutation $[6, 5, 3, 4, 2, 1, 8, 7, 10, 9]$. A *genome* is a set of chromosomes. Given a genome $\Pi = \{\pi(1), \ldots, \pi(N)\}$ containing N chromosomes, there are 2^N distinct ways for representing this genome based on the two representations of chromosomes.

Let $\pi = [\pi_1, \ldots, \pi_{i-1}, \pi_i, \ldots, \pi_j, \pi_{j+1}, \ldots, \pi_n]$ be a chromosome and $1 \leq i \leq j \leq n$. A *signed reversal* $\rho(\pi, i, j)$ is a rearrangement event that transforms the chromosome π into the chromosome $[\pi_1, \ldots, \pi_{i-1}, -\pi_j, \ldots, -\pi_i, \pi_{j+1}, \ldots, \pi_n]$. A *fission* $\rho(\pi, i)$ for $1 < i \leq n$ is the rearrangement event that "breaks" the chromosome π into two chromosomes $[\pi_1, \ldots, \pi_{i-1}]$ and $[\pi_i, \ldots, \pi_n]$. Given the chromosomes $\pi = [\pi_1, \ldots, \pi_n]$ and $\sigma = [\sigma_1, \ldots, \sigma_n]$, a *fusion* $\rho(\pi, \sigma)$ transforms chromosomes π and σ into the chromosome $[\pi_1, \ldots, \pi_n, \sigma_1, \ldots, \sigma_n]$. Notice that Hannenhalli and Pevzner [6] define fissions and fusions based on the definition of translocations. Because we do not use translocations, we use direct, equivalent definitions instead.

The second model for genomes, chromosomes, and genes is called the *algebraic formalism* [14, 16, 9]. In the classical formalism, a chromosome α is a mapping of a position i into the gene α_i. This mapping representation follows straightforwardly the structure of a linear chromosome; that is, a sequence of genes. In the algebraic formalism, we represent a chromosome not as a sequence, but as a cycle, that is a circular list of genes. For example, given the representation $\alpha = [\alpha_1, \ldots, \alpha_n]$, instead of being a sequence representing a linear chromosome, we can view it as a permutation that maps α_i into α_{i+1} for $1 \leq i < n$ and α_n into α_1 and α models a circular chromosome. Fig. 1 illustrates the distinction between the two kinds of representation. Different representations for the same genome imply distinct interpretations for data structures that implement these representation. We will show later that we can take advantage of this distinct interpretation of algebraic formalism to implement rearrangement events faster than in the classical formalism.

We now formalize and present the same basic concepts for modeling genomes in the algebraic formalism.

Given a permutation π over the set E, the *orbit* of $x \in E$ under the permutation π, denoted by $orb(\pi, x)$, is the set $\{y \mid y = \pi^k x \text{ for an integer } k\}$. An orbit is called *nontrivial* when it has more than one element. Let $o(\pi, E)$ be the number of orbits in permutation π. The *support* of a permutation π over E is the set $Supp(\pi) = \{x \in E \mid \pi x \neq x\}$. A *cycle* is a permutation α over E such that it has at most one nontrivial orbit. A cycle α is an *r-cycle* when its nontrivial orbit contains $r > 1$ elements or an 1-*cycle* when $\alpha = \iota$, where ι is the permutation such that $\iota x = x$ for any $x \in E$. As we have mentioned previously, a cycle α can be represented as a circular list of elements. For instance, let α be a cycle that maps a into b, b into c, c into a, and d into d over the set $E = \{a, b, c, d\}$. The cycle α is represented by $(a\ b\ c)$. The *product* of permutation β by α both over E, denoted by $\alpha\beta$, is the permutation that maps x to $\alpha(\beta x)$ for any $x \in E$. A *k-cycle decomposition* of a permutation π is a representation of π as a product of k-cycles, not necessarily disjoint. The *norm* of π, denoted by $\|\pi\|$, is the minimum number of 2-cycles whose product is π.



(a)

1	2	3	4	5	−1	−2	−3	−4	−5
−2	−1	−5	1	−3	−4	3	2	5	4

(b)

1	2	3	4	5	6	7	8	9	10
6	5	3	4	2	1	8	7	9	10

(c)

Figure 1: (a) Circular chromosome. (b) Array data structure for Algebraic formalism representation: $\alpha = (-3\ 2\ -1\ -4\ 5)(-5\ 4\ 1\ -2\ 3)$ over the $E = \{x \mid 1 \leq |x| \leq 5\}$. (c) Array data structure for Classical representation: $\beta = [6,\ 5,\ 3,\ 4,\ 2,\ 1,\ 8,\ 7,\ 9,\ 10]$ is the image of the chromosome $[-3,\ 2,\ -1,\ -4,\ 5]$. In the classical representation, we suppose that the extremities are -3 and 5.

A DNA chromosome has two strands with complementary orientation. A set of genes E and a permutation Γ over E is called a *gene system*, denoted by (E, Γ), when $Supp(\Gamma) = E$ and $\Gamma\Gamma = \iota$. The permutation Γ associates each gene to its complementary. For instance, given the set of genes $E = \{-4,\ -3,\ -2,\ -1,\ 1,\ 2,\ 3,\ 4\}$ and the permutation $\Gamma = (1\ -1)(2\ -2)(3\ -3)(4\ -4)$, the pair (E, Γ) is a gene system since the support of Γ is E and $\Gamma\Gamma x = x$ for any $x \in E$. A partition of E into two sets E_+ and E_- such that $E_- = \{\Gamma x \mid x \in E_+\}$ is called a *valid partition*. The valid partition where all the the genes in E_+ (resp. in E_-) have the same orientation is called the *oriented partition* of E. For instance, the oriented partition of $E = \{-5,\ -4,\ -3,\ -2,\ -1,\ 1,\ 2,\ 3,\ 4,\ 5\}$ is $E_+ = \{1,\ 2,\ 3,\ 4,\ 5\}$ and $E_- = \{-1,\ -2,\ -3,\ -4,\ -5\}$.

Given a gene system (E, Γ), a cycle α is called a *strand* when $x \in Supp(\alpha)$ implies $\Gamma x \notin Supp(\alpha)$ for each $x \in E$. The *conjugation* of a permutation α by β, denoted by $\beta \cdot \alpha$, is $\beta\alpha\beta^{-1}$ [14]. A *chromosome* is a product of two strands α and $\Gamma \cdot \alpha^{-1}$. Two chromosomes are *disjoint* when their supports are disjoint. A *genome* is a product of disjoint chromosomes. A fundamental property of genomes is: if π is a genome, then $\Gamma\pi\Gamma = \pi^{-1}$.

We deal in this work with genomes composed solely of circular chromosomes instead of linear chromosomes since circular chromosomes are naturally modeled by cycles. In some cases, it is straightforward to translate problems in circular genomes to liner genomes and vice versa [8, 15]. In a recent work, Feijao and Meidanis [5] extended the algebraic formalism to model circular, as well as linear chromosomes by interpreting some of its core concepts in a original way.

Given a genome π in the gene system (E, Γ), the permutation $(u\ v)(\pi\Gamma u\ \pi\Gamma v)$ is called a 2-*break* applicable to π when $u \neq v$ and $u, v \in E$. It is easy to see that $\rho = (u\ v)(\pi\Gamma u\ \pi\Gamma v)$ is a 2-break if and only if $\rho\pi$ is a genome in (E, Γ). A 2-break is classified as a fission, a fusion, or signed reversal depending on the distribution of the elements u, v, $\pi\Gamma u$, and $\pi\Gamma v$ among the orbits of π. We introduce a few concepts that will be used to better explain the relation between a 2-break applicable to π and the distribution of the elements in its support among the orbits of π. Given the permutations α and β, both over a set E, we say that α *divides* β, denoted by $\alpha|\beta$, when $\|\beta\alpha^{-1}\| = \|\beta\| - \|\alpha\|$. Meidanis and Dias [14] point out that a 2-cycle $(a\ b)$ divides a permutation β if and only if the elements a and b belong to the same orbit in β. Moreover, a product of a 2-cycle $\alpha = (a\ b)$ and a permutation β results in the permutation $\alpha\beta$ such that a and b belong to distinct orbits in $\alpha\beta$ if and only if a and b belong to the same orbit in β. In other words, the product by α "breaks" the cycle in β whose orbit contains a and b into two cycles in $\alpha\beta$, or it "joins" the two cycles, the orbit of each one containing one of a or b. But this property is a description of the action of fissions and fusions on genomes! Fig. 2 illustrates the action of 2-breaks on genomes as 2-cycles operating on cycles.

Using divisibility, we can classify 2-breaks as fissions, fusions, and signed reversals as follows:

Definition 1. *Let π be a genome in the gene system (E, Γ), and $\rho = (u\ v)(\pi\Gamma v\pi\Gamma u)$ a 2-break applicable to π. Then*

1. *Operation ρ is a* fission *on π when u and v belong to the same orbit of π, that is, $(u\ v)|\pi$.*

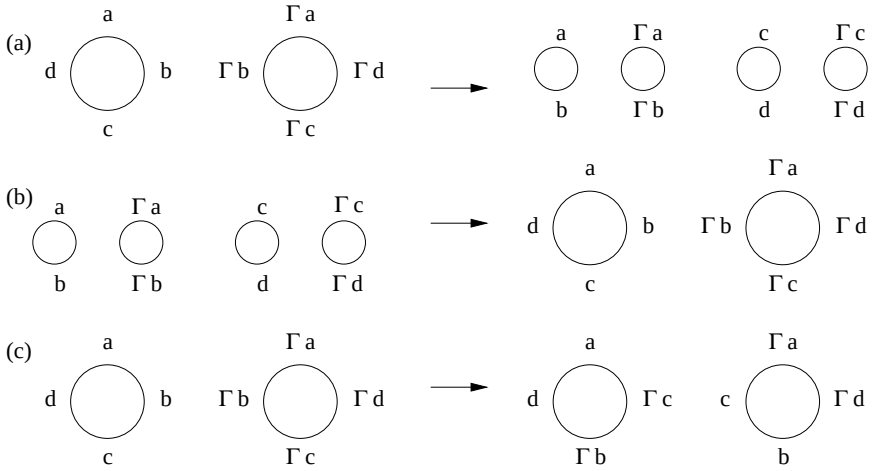


Figure 2: Genome rearrangement events as breaking-joining operation on cycles. All the cycles are clockwise oriented. (a) The genome $\pi = (a\ b\ c\ d)(\Gamma d\ \Gamma c\ \Gamma b\ \Gamma a)$ is transformed into the genome $\rho\pi = (a\ b)(\Gamma b\ \Gamma a)(c\ d)(\Gamma d\ \Gamma c)$ by the fission $\rho = (a\ c)(\Gamma b\ \Gamma d)$. (b) The genome $\pi = (a\ b)(\Gamma b\ \Gamma a)(c\ d)(\Gamma d\ \Gamma c)$ is transformed into the genome $\rho\pi = (a\ b\ c\ d)(\Gamma d\ \ \Gamma c\ \Gamma b\ \Gamma a)$ by the fusion $\rho = (a\ c)(\Gamma d\ \Gamma b)$. (c) The genome $\pi = (a\ b\ c\ d)(\Gamma d\ \Gamma c\ \Gamma b\ \Gamma a)$ is transformed into the genome $\rho\pi = (a\ \Gamma c\ \Gamma b\ d)(\Gamma d\ b\ c\ \Gamma a)$ by the signed reversal $\rho = (b\ \Gamma b)(d\ \Gamma a)$.

2. *Operation ρ is a* signed reversal on π *when u and v belong to distinct orbits of the same chromosome.*

3. *Operation ρ is a* fusion on π *when u and v belong to orbits of distinct chromosomes.*

Given the genomes π and σ in the gene system (E, Γ), the *algebraic rearrangement by fusions, fissions, and signed reversals problem* consists of finding a sequence $\rho_1,\ \ldots,\ \rho_k$ such that:

$$\sigma = \rho_k\ \rho_{k-1}\ \ldots\ \rho_1\ \pi$$

where the rearrangement event ρ_{i+1} is a fusion, fission, or signed reversal on $\rho_i\ \rho_{i-1}\ \ldots\ \rho_1\ \pi$, for $1 \leq i \leq k$ and k is minimum. We call this minimum k the genomic distance $d(\pi,\ \sigma)$.

As we are going to show in the next sections, the algebraic formalism allows one to implement simple and efficient data structures that represent genomes and chromosomes as cycles. We propose an algorithm based on these efficient data structures for the comparison of genomes based on 2-breaks, that is, signed reversals, fissions, and fusions. These rearrangement events, especially signed reversals [1, 7, 6, 2, 12], have been shown to be important in comparative analysis of genomes [18, 4].

The paper is organized as follows. In Section 2 we show how to detect 2-breaks belonging to a minimum sequence of rearrangement events that transforms a genome into another. In Section 3 we present data structures that implement the main ideas behind the algebraic formalism and we design a polynomial time algorithm for finding a minimum rearrangement event sequence that transforms a genome into another. Moreover, we present a formula for the genomic distance based on the norm of a permutation. We summarize the results in Section 4.

2 Good Events

Given the genomes π and σ in the gene system (E, Γ) and a sequence of 2-breaks $\rho_1, \ldots, \rho_k$ such that $\rho_k \ldots \rho_1\pi = \sigma$ and ρ_{i+1} is applicable to $\rho_i \ldots \rho_1\pi$ for $1 \leq i \leq k-1$, we have $\rho_k \ldots \rho_1 = \sigma\pi^{-1}$. Since the permutation $\sigma\pi^{-1}$ can be rewritten as a product of 2-breaks then the algebraic rearrangement by fusions, fissions, and signed reversals problem is equivalent to find a product $\rho_k \ldots \rho_1$ that equals to $\sigma\pi^{-1}$ and each ρ_{i+1} is a 2-break applicable to $\rho_i \ldots \rho_1\pi$ for $1 \leq i \leq k-1$ and k is minimum. We define some concepts and show some properties of the permutation $\sigma\pi^{-1}$ that will help find one such minimum sequence of 2-breaks.

Let θ be a permutation over a set E. A cycle α is said to be *a cycle of* θ when α is one of the cycles in the unique disjoint cycle decomposition of θ. A *pair* of $\sigma\pi^{-1}$ is a couple of cycles α and $(\pi\Gamma) \cdot \alpha^{-1}$ of $\sigma\pi^{-1}$. Let $c(\pi, \sigma)$ be the number of pairs of $\sigma\pi^{-1}$. It has been shown that the number of pairs of $\sigma\pi^{-1}$ is $c(\pi, \sigma) = (|E| - \|\sigma\pi^{-1}\|)/2$ [14]. We denote $c(\rho\pi, \sigma) - c(\pi, \sigma)$ by $\Delta c(\rho, \pi, \sigma)$ where ρ is a 2-break applicable to π. The concept of a pair of $\sigma\pi^{-1}$ corresponds to the concept of *alternate cycles* in the cycle diagram (or breakpoint graph) of a genome [6, 7, 12]. If ρ is a signed reversal on π, then we have $\Delta c(\rho, \pi, \sigma) \in \{-1, 0, 1\}$ [7].

A *good event* for (π, σ) is a fusion, fission, or a signed reversal ρ such that $\Delta c(\rho, \pi, \sigma) = 1$.

Lemma 1. *Given genomes π, σ in the gene system (E, Γ), a 2-break ρ applicable to π is a good event for (π, σ) if and only if $\rho|\sigma\pi^{-1}$.*

Proof. If $\rho|\sigma\pi^{-1}$ then $\|\sigma\pi^{-1}\rho^{-1}\| = \|\sigma\pi^{-1}\| - \|\rho\|$. Manipulating the later formula:

$$\begin{aligned} \frac{\|\rho\|}{2} &= \frac{\|\sigma\pi^{-1}\| - \|\sigma\pi^{-1}\rho^{-1}\|}{2} \\ &= \frac{|E| - \|\sigma\pi^{-1}\rho^{-1}\| - |E| + \|\sigma\pi^{-1}\|}{2} \\ &= c(\rho\pi, \sigma) - c(\pi, \sigma), \end{aligned}$$

and since $c(\rho\pi, \sigma) - c(\pi, \sigma) = \Delta c(\rho, \pi, \sigma)$ then $\Delta c(\rho, \pi, \sigma) = \frac{\|\rho\|}{2}$. Since ρ is a 2-break then $\frac{\|\rho\|}{2} = 1$. Therefore ρ is a good event for (π, σ).

Conversely, if ρ is a good event for (π, σ) then $\Delta c(\rho, \pi, \sigma) = \frac{\|\rho\|}{2}$, that is, we have $c(\rho\pi, \sigma) - c(\pi, \sigma) = \|\rho\|/2$. By definition of $c(\pi, \sigma)$ we have

$$\frac{\|\sigma\pi^{-1}\| - \|\sigma\pi^{-1}\rho^{-1}\|}{2} = \frac{\|\rho\|}{2}.$$

Therefore $\|\sigma\pi^{-1}\rho^{-1}\| = \|\sigma\pi^{-1}\| - \|\rho\|$ and hence $\rho|\sigma\pi^{-1}$. □

Lemma 2. *Given genomes π, σ in the gene system (E, Γ), for any sequence of rearrangement events $\rho_1, \ldots, \rho_k$, such that $\rho_k \ldots \rho_1\pi = \sigma$ and ρ_i is applicable to the genome $\rho_{i-1} \ldots \rho_1\pi$, we have:*

1. $k \geq \frac{\|\sigma\pi^{-1}\|}{2}$;

2. $k = \frac{\|\sigma\pi^{-1}\|}{2}$ *if and only if each* ρ_i *is a good event for* $(\rho_{i-1} \dots \rho_1\pi, \sigma)$ *for* $1 \leq i \leq k$.

Proof. 1. Let $\rho_1, \dots, \rho_k$ be a sequence of rearrangement events such that $\rho_k \dots \rho_1\pi = \sigma$ and ρ_i is applicable to $\rho_{i-1} \dots \rho_1\pi$ for $1 \leq i \leq k$. Therefore $\rho_k \dots \rho_1 = \sigma\pi^{-1}$, and we get the following upper bound for $\|\sigma\pi^{-1}\|$.

$$\begin{aligned} \|\sigma\pi^{-1}\| &= \|\rho_k \dots \rho_1\| \\ &\leq \|\rho_k\| + \dots + \|\rho_1\| \\ &= 2\sum_{j=1}^{k} \frac{\|\rho_j\|}{2} \\ &= 2k \end{aligned}$$

Therefore we have $k \geq \frac{\|\sigma\pi^{-1}\|}{2}$

2. Firstly, we prove the "if" part, that is, we assume that each rearrangement event ρ_i is a good event for $(\rho_{i-1} \dots \rho_1\pi, \sigma)$ for $1 \leq i \leq k$. By definition of 2-breaks and norm, we have $\|\rho\| = 2$, where ρ is a 2-break. Therefore, we have:

$$k = \frac{\|\rho_1\| + \dots + \|\rho_k\|}{2}. \tag{1}$$

Since the rearrangement event ρ_i is a good event for $(\rho_{i-1} \dots \rho_1\pi, \sigma)$ for $1 \leq i \leq k$ then we have

$$c(\rho_i \dots \rho_1\pi, \sigma) - c(\rho_{i-1} \dots \rho_1\pi, \sigma) = 1.$$

By definition of number of pairs $c(,)$ we have

$$\frac{\|\sigma\pi^{-1}\rho_1^{-1} \dots \rho_{i-1}^{-1}\|}{2} + \frac{\|\sigma\pi^{-1}\rho_1^{-1} \dots \rho_i^{-1}\|}{2} = 1. \tag{2}$$

Using $\|\rho\|/2 = 1$ for any 2-break ρ, Equation 1, Equation 2, and some manipulation:

$$\begin{aligned} k &= \frac{\|\sigma\pi^{-1}\| - \|\sigma\pi^{-1}\rho_1^{-1} \dots \rho_k^{-1}\|}{2} \\ &= \frac{\|\sigma\pi^{-1}\| - \|\sigma\sigma^{-1}\|}{2} \\ &= \frac{\|\sigma\pi^{-1}\|}{2}. \end{aligned}$$

Therefore $k = \frac{\|\sigma\pi^{-1}\|}{2}$.

On the other hand, if $k = \sum_{j=1}^{k} \frac{\|\rho_j\|}{2}$ then $\|\sigma\pi^{-1}\| = \sum_{j=1}^{k} \|\rho_j\|$.

By the triangular inequality property, we have

$$\|\sigma\pi^{-1}\rho_1^{-1} \dots \rho_{i-1}^{-1}\| \leq \|\sigma\pi^{-1}\rho_1^{-1} \dots \rho_i^{-1}\| + \|\rho_i\|,$$

for $1 \leq i \leq k$, in other words we have

$$0 \leq \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_i^{-1}\| - \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}\| + \|\rho_i\|,$$

for $1 \leq i \leq k$. Since each term $\|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_i^{-1}\| - \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}\| + \|\rho_i\|$ is non negative and manipulating the expanded sum we get

$$\begin{aligned} 0 \leq & \sum_{i=1}^{k} \Big(\|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_i^{-1}\| - \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}\| + \|\rho_i\| \Big) \\ = & \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_k^{-1}\| - \|\sigma\pi^{-1}\| + \sum_{i=1}^{k} \|\rho_i\|. \end{aligned}$$

But since $\sigma\pi^{-1}\rho_1^{-1}\dots\rho_k^{-1} = \iota$ and $\sum_{i=1}^{k} \|\rho_i\| = \|\sigma\pi^{-1}\|$ then the sum

$$\sum_{i=1}^{k} \Big(\|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_i^{-1}\| - \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}\| + \|\rho_i\| \Big)$$

is zero.

Then $\|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_i^{-1}\| - \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}\| + \|\rho_i\| = 0$ for $1 \leq i \leq k$; i.e. we have

$$\|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_i^{-1}\| = \|\sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}\| - \|\rho_i\|$$

for $1 \leq i \leq k$. Therefore, we have $\rho_i | \sigma\pi^{-1}\rho_1^{-1}\dots\rho_{i-1}^{-1}$ for $1 \leq i \leq k$ and by Lemma 1 each ρ_i applicable to $\rho_{i-1}\dots\rho_1\pi$ is a good event for $(\rho_{i-1}\dots\rho_1\pi, \sigma)$.

□

Lemma 3. *Given any distinct genomes π, σ in the gene system (E, Γ), there is a 2-break ρ applicable to π such that ρ is a good event for (π, σ).*

Proof. Consider the 2-break $\rho = (x\ \sigma\pi^{-1}x)(\pi\Gamma\sigma\pi^{-1}x\ \pi\Gamma x)$ where x and $\sigma\pi^{-1}x$ are elements of E. There are three cases depending on which orbits these elements belong to in π:

- If $\sigma\pi^{-1}x \in orb(\pi, x)$ then $\rho = (x\ \sigma\pi^{-1}x)(\pi\Gamma\sigma\pi^{-1}x\ \pi\Gamma x)$ is a fission on π since we have $(x\ \sigma\pi^{-1}x)|\pi$.
- If $\sigma\pi^{-1}x \notin orb(\pi, x)$ and x and $\sigma\pi^{-1}x$ belong to the orbits of the same chromosome then we are going to show that the 2-break $\rho = (x\ \sigma\pi^{-1}x)(\pi\Gamma\sigma\pi^{-1}x\ \pi\Gamma x)$ is a signed reversal on π and it is a good event for (π, σ). Elements x and $\sigma\pi^{-1}x$ belong to distinct strands of the genome π and we have $\Gamma\pi^{-1}\sigma\pi^{-1}x = \pi\Gamma\sigma\pi^{-1}x$ by the fundamental property of genomes $\Gamma\pi\Gamma = \pi^{-1}$, so $\pi\Gamma\sigma\pi^{-1}x \in orb(\pi, x)$ and therefore $(x\ \pi\Gamma\sigma\pi^{-1}x)|\pi$. In addition, we have $x \neq \pi\Gamma\sigma\pi^{-1}x$ because otherwise $x = \pi\Gamma\sigma\pi^{-1}x$ implies $\pi\Gamma x = \sigma\Gamma\pi\Gamma x$ and then there is an element $z \in E$ such that $\sigma\Gamma z = z$, that is, two complementary genes belong to the same strand in σ, which contradicts the definition of a genome in the gene system (E, Γ). Therefore, the rearrangement event $(x\ \sigma\pi^{-1}x)(\pi\Gamma\sigma\pi^{-1}x\ \pi\Gamma x)$ is a signed reversal on π.

- If x and $\sigma\pi^{-1}x$ belong to orbits of distinct chromosomes of π then rearrangement event $\rho = (x\ \sigma\pi^{-1}x)(\pi\Gamma\sigma\pi^{-1}x\ \pi\Gamma x)$ is a fusion on π since $(x\ \sigma\pi^{-1}x) \nmid \pi$.

Since $\sigma\pi^{-1}$ is a product of pairs $\alpha\pi\Gamma\alpha^{-1}\pi\Gamma$ then for any cycle $(a_1\ \ldots\ a_m)$ of $\sigma\pi^{-1}$ there is a cycle $(\pi\Gamma a_m\ \ldots\ \pi\Gamma a_1)$ of $\sigma\pi^{-1}$, and therefore $\pi\Gamma x \notin orb(\sigma\pi^{-1}, x)$. Moreover, because $\sigma\pi^{-1}x \in orb(\sigma\pi^{-1}, x)$, $\pi\Gamma\sigma\pi^{-1}x \in orb(\sigma\pi^{-1}, \pi\Gamma x)$, and $\pi\Gamma x \notin orb(\sigma\pi^{-1}, x)$ then $\rho|\sigma\pi^{-1}$, and by Lemma 1 the 2-break ρ is a good event for (π, σ). □

3 Algorithm

In this section we present an algorithm for finding a sequence of good events that transform the genome π into σ both in the gene system (E, Γ), which takes $O(nh)$ running time, where n is the number of genes and h is the time spent to convert a gene name into an integer. We describe a data structure to represent the genome π and the permutation $\sigma\pi^{-1}$ that allows us to perform queries, fissions, fusions, and signed reversals each in $O(h)$ running time. The data structure is a simple combination of arrays that models the mapping of genomes π, π^{-1}, and σ; and a hash table and a list of circular lists representing the permutation $\sigma\pi^{-1}$.

3.1 Genes in $\{-n,\ \ldots,\ -1,\ 1,\ \ldots,\ n\}$

We firstly treat the case in which π and σ are genomes in the gene system (E, Γ) where $E = \{-n,\ \ldots,\ -1,\ 1,\ \ldots,\ n\}$ and $\Gamma = (1\ \ -1)(2\ \ -2)\ldots(n\ \ -n)$. In this case, the running time is $O(n)$. Later, we extend the solution for gene systems whose set E contains gene names, which require hash look-ups. Genomes π and σ are represented by arrays P and S where $P[x] = \pi x$ and $S[x] = \sigma x$ for $x \in E$. Similarly we define the array $invP$ where $invP[x] = \pi^{-1}x$ for $x \in E$. The complementary gene Γx is simply $-x$. Moreover, we describe the permutation $\sigma\pi^{-1}$ as a doubly linked list Q whose nodes that we call *pairs* contain pointers to two circular lists. Each node in the Q list models a pair of cycles α and β of $\sigma\pi^{-1}$ where $\beta = (\pi\Gamma)\cdot\alpha^{-1}$ and these cycles are the circular lists pointed to by the node. We do not represent 1-cycles in the list (we call a pair with two 1-cycles a *trivial pair*). Finally, we use an array T whose entry $T[x].node$ points to the node in Q that contains x and $T[x].pair$ points to the pair node of x, for every $x \in E$ (we use the convention that $T[x].node = T[x].pair = NIL$ when there is no x in Q). Fig. 3 illustrates the data structure for a pair of genomes in a gene system with eight genes.

The array P is initialized by assigning $P[x] = \pi x$, for each gene x in E. The same procedure can be used to initialize the array S. We can initialize $invP$ at the same time we initialize P by assigning $invP[\pi x] = x$ for each assignment of x in P. The initialization of Q is based on the previously initialized arrays S and $invP$. At the beginning of the procedure, all pointers in T are initialized to NIL. For each gene $x \in E$, if $S[invP[x]] \neq x$ and $T[x].node = NIL$, then the procedure *buildPair(x)* is called. The procedure buildPair(x) is responsible for creating a new pair with x and including it in Q. The procedure starts creating a pair N, storing the value of x in auxiliary variable y, and entering in a loop that creates nodes for y and $\pi\Gamma y$, updates the pointers in $T[y]$ and $T[P[-y]]$ to link to the pair N and the nodes containing y and $\pi\Gamma y$ respectively, attaches the nodes to the circular lists

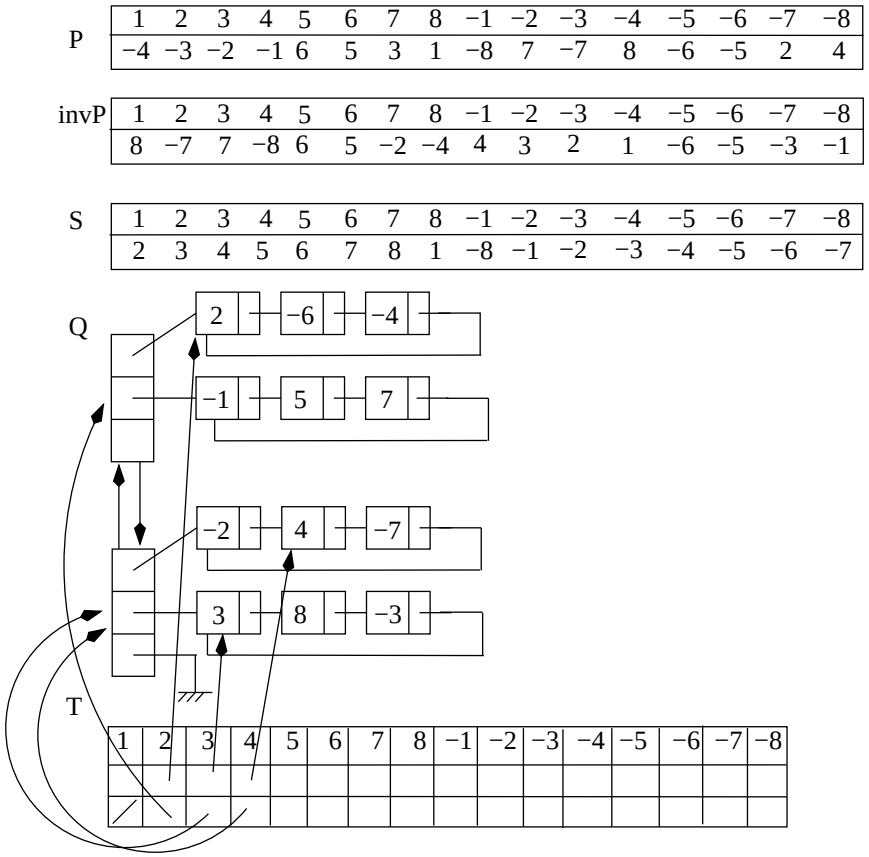


Figure 3: Genomes $\pi = (3\ -2\ 7)(-7\ 2\ -3)(-5\ -6)(6\ 5)(1\ -4\ 8)(-8\ 4\ -1)$ and $\sigma = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)(-8\ -7\ -6\ -5\ -4\ -3\ -2\ -1)$ in the gene system (E, Γ) where $E = \{x \mid |x| \in [8]\}$ and $\Gamma x = -x$ are represented by arrays P and S. The permutation $\sigma\pi^{-1}$ is represented by the list Q. The array T makes an entry indexed by a gene x to point to the node containing x and to the pair of circular lists in Q. In order to make the figure clear, we show the pointers of the first four genes of T only.

being constructed, and assigns $S[invP[y]]$ to y until $y = x$. Suppose that the pair N in Q models the permutation $\alpha(\pi\Gamma) \cdot \alpha^{-1}$ and $y \in Supp(\alpha)$, so y is mapped to $\sigma\pi^{-1}y$ in α while $\pi\Gamma\sigma\pi^{-1}y$ is mapped to $\pi\Gamma y$ in $(\pi\Gamma) \cdot \alpha^{-1}$. Therefore the node containing y has to be attached to the tail of the circular list modeling α and the node containing $\pi\Gamma y$ has to be attached to the head of the circular list modeling $(\pi\Gamma) \cdot \alpha^{-1}$. After the end of the loop, both lists are closed. The procedure that initializes Q, called *initializePairList(Q)*, and the procedure that builds a new pair and insert it into Q are illustrated by Fig. 4 and Fig. 5 respectively.

```
initializePairsList(Pair List Q, Array S, Array
invP, Array T)
1.    for each x ∈ E do
2.        T[x].node = T[x].pair = NIL;
3.    for each x ∈ E do
4.        if S[invP[x]] ≠ x
5.            if T[x].node = NIL
6.                buildPair(x);
7.    return Pair List Q;
```

Figure 4: Procedure **initializePairsList** used to initialize Pairs List Q.

The arrays P, $invP$, S, T, and the list Q provide the following operations: query, exchange, remove, trivial pair testing, and trivial pair deletion. The fission, fusion, and signed reversal events combine the previous operations. A *query* for an element $x \in E$ in genome π (or σ) over E consists of finding πx (or σx). The query of an element $x \in E$ in array A (array A can be P, S, or $invP$) is implemented by retrieving $A[x]$. An *exchange* of two entries $A[x]$ and $A[y]$ (array A can be P, S, or $invP$) consists of exchanging the contents of $A[x]$ and $A[y]$. The *remove* operation, denoted by $remove(Q, x, y)$, removes the nodes that contains the genes $x, y \in E$. As the circular lists shrink during the execution of the algorithm, eventually they become nodes of a trivial pair and need to be removed from Q. The *trivial pair testing* operation, denoted by the procedure $trivialPair(Q, x)$, is responsible to detect whether the pair containing x is a trivial pair. The trivial pair testing is implemented by verifying whether the content of the first node has the same content of its next node in one of the cycles of the pair. The *trivial pair deletion* operation deletes a trivial pair from Q that contains the genes $x, \pi\Gamma x \in E$. The trivial pair deletion operation, called $deletePair(pair)$, is implemented by using the pointer $T[x].pair$ and the standard removal of nodes from a doubly linked list and it involves also a modification of the pointers

buildPair(gene x)

1. Create pair N and insert N into Q;
2. $y = x$;
3. *do*
4. Create new node A with data y;
5. Attach A to the tail of circular list 1 of N;
6. $T[y].node = A$;
7. $T[y].pair = N$;
8. Create node B with data $P[-y]$;
9. Attach B to the head of the circular list 2 of N;
10. $T[P[-y]].node = B$;
11. $T[P[-y]].pair = N$;
12. $y = S[invP[y]]$;
13. *until* $y = x$;
14. Close circular list containing x;
15. Close circular list containing $P[-x]$;

Figure 5: Procedure **buildPair** used to build a pair N and include it into Q.

node and *pair* in $T[x]$ and $T[\pi\Gamma x]$ to NIL, where x and $\pi\Gamma x$ are the genes in the trivial pair.

We discuss now how to implement fissions, fusions, and signed reversals using the previous operations. Given π and σ in (E, Γ), we restrict the implementation to 2-breaks of the form is $(a\ \sigma\pi^{-1}a)(\pi\Gamma\sigma\pi^{-1}a\ \pi\Gamma a)$. All such events are good events for (π, σ). We make such a restriction to guarantee that each rearrangement event can be performed in constant running time. The 2-break $\rho = (a\ \sigma\pi^{-1}a)(\pi\Gamma\sigma\pi^{-1}a\ \pi\Gamma a)$ applicable to genome π is performed as follows:

1. Querying genes. The elements a, $\sigma\pi^{-1}a$, $\pi\Gamma\sigma\pi^{-1}a$, and $\pi\Gamma a$ in E are queried in order to find the elements $\pi^{-1}a$, $\pi^{-1}\sigma\pi^{-1}a$, $\pi^{-1}\pi\Gamma\sigma\pi^{-1}a$, and $\pi^{-1}\pi\Gamma a$ in $invP$.

2. Updating the array P. The elements $\pi^{-1}a$, $\pi^{-1}\sigma\pi^{-1}a$, $\pi^{-1}\pi\Gamma\sigma\pi^{-1}a$, and $\pi^{-1}\pi\Gamma a$ are mapped to $\sigma\pi^{-1}a$, a, $\pi\Gamma a$, and $\pi\Gamma\sigma\pi^{-1}a$ respectively in $\rho\pi$ while the remaining genes do not change their mappings because in a product of permutation $\alpha\beta$ the element $\beta^{-1}x$ is mapped to an element y different from x when x belongs to the support of α, otherwise $\alpha x = x$ and $\alpha\beta\beta^{-1}x = x$. So, the genes $\pi^{-1}a$, $\pi^{-1}\sigma\pi^{-1}a$, $\pi^{-1}\pi\Gamma\sigma\pi^{-1}a$, and $\pi^{-1}\pi\Gamma a$ in P are updated to $\sigma\pi^{-1}a$, a, $\pi\Gamma a$, and $\pi\Gamma\sigma\pi^{-1}a$ respectively in P', where P' represents $\rho\pi$. Since $P[\pi^{-1}a] = a$ and $P[\pi^{-1}\sigma\pi^{-1}a] = \sigma\pi^{-1}a$ then we can exchange the contents of $P[\pi^{-1}a]$ and $P[\pi^{-1}\sigma\pi^{-1}a]$ to obtain the correct mappings of $\pi^{-1}a$ and $\pi^{-1}\sigma\pi^{-1}a$ in P'. Similarly, we have $P[\pi^{-1}\pi\Gamma\sigma\pi^{-1}a] = \pi\Gamma\sigma\pi^{-1}a$ and $P[\pi^{-1}\pi\Gamma a] = \pi\Gamma a$, so exchanging $P[\pi^{-1}\pi\Gamma\sigma\pi^{-1}a]$ and $P[\pi^{-1}\pi\Gamma a]$ gives the mappings of $\Gamma\sigma\pi^{-1}a$ and Γa in P'.

3. Updating the array $invP$. The elements a, $\sigma\pi^{-1}a$, $\pi\Gamma\sigma\pi^{-1}a$, and $\pi\Gamma a$ in E are mapped to $\pi^{-1}\sigma\pi^{-1}a$, $\pi^{-1}a$, $\pi^{-1}\pi\Gamma a$, and $\pi^{-1}\pi\Gamma\sigma\pi^{-1}a$ respectively in $\pi^{-1}\rho^{-1}$ (the inverse of $\rho\pi$) while the remaining genes do not change their mappings for the same reasons presented before. The genes in $invP[a]$, $invP[\sigma\pi^{-1}a]$, $invP[\pi\Gamma\sigma\pi^{-1}a]$, and $invP[\pi\Gamma a]$ must be updated to $\pi^{-1}\sigma\pi^{-1}a$, $\pi^{-1}a$, $\pi^{-1}\pi\Gamma a$, and $\pi^{-1}\pi\Gamma\sigma\pi^{-1}a$ respectively in $invP'$, where the array $invP'$ represents the genome $\pi^{-1}\rho^{-1}$. Since we have $invP[a] = \pi^{-1}a$ and $invP[\sigma\pi^{-1}a] = \pi^{-1}\sigma\pi^{-1}a$ then $invP$ is updated to $invP'$ by exchanging the genes $invP[a]$ and $invP[\sigma\pi^{-1}a]$. Similarly, we have the following mappings $invP[\pi\Gamma\sigma\pi^{-1}a] = \pi^{-1}\pi\Gamma\pi^{-1}a$ and $invP[\pi\Gamma a] = \pi^{-1}\pi\Gamma a$, so exchanging $invP[\pi\Gamma a]$ and $invP[\pi\Gamma\sigma\pi^{-1}a]$ gives the mappings of $\pi\Gamma a$ and $\pi\Gamma\sigma\pi^{-1}a$ in $invP'$.

4. Updating the list Q. The list Q is updated by a removal of the elements $\sigma\pi^{-1}a$ and $\pi\Gamma a$ from their corresponding circular lists. Removing $\sigma\pi^{-1}a$ and $\pi\Gamma a$ can be done in constant time since both elements are in the next nodes of a and $\pi\Gamma\sigma\pi^{-1}a$ respectively in the circular list and a and $\pi\Gamma\sigma\pi^{-1}a$ are directly accessed using $T[a].node$ and $T[\pi\Gamma\sigma\pi^{-1}a].node$. If it remains a single element in each circular list pointed by a node of Q then this pair is a trivial pair and it is removed using a as a parameter to the pair deletion procedure.

Fig. 6 illustrates the action of a 2-break on the arrays P and $invP$.

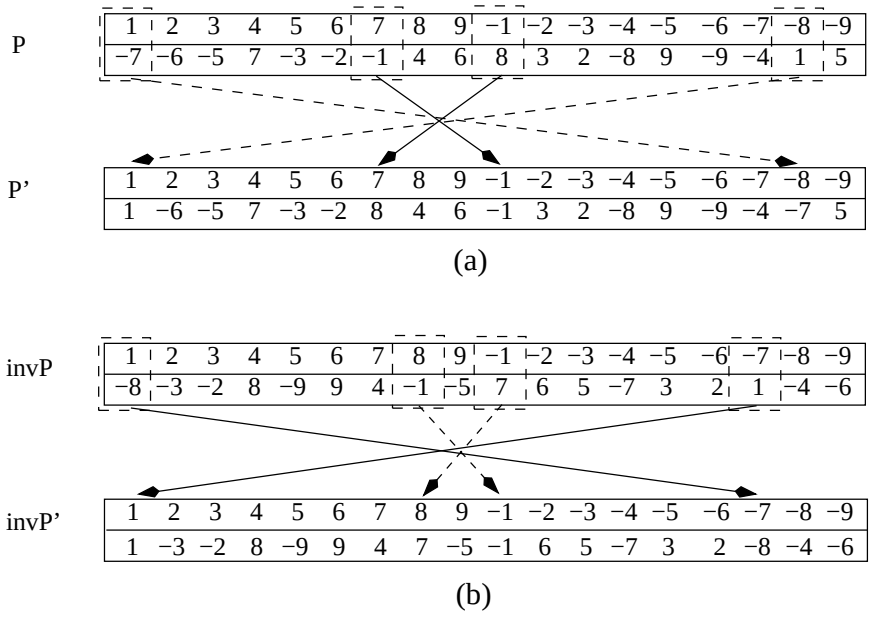


Figure 6: Example of the action of a 2-break on arrays P and $invP$. The arrays P and $invP$ are based on the genome $\pi = (-3\ 2\ -6\ -9\ 5)(-5\ 9\ 6\ -2\ 3)(-7\ -4\ -8\ 1)(-1\ 8\ 4\ 7)$ in the gene system (E, Γ). The arrays P' and $invP'$ represent the genome $\rho\pi$ where $\rho = (1\ -7)(-1\ 8)$ is a 2-break applicable to π. (a) The genes $P[7]$ and $P[-1]$ are exchanged, as well as the content of the pair $P[1]$ and $P[-8]$ in the array P. (b) In the array $invP$, the genes $invP[1]$ and $invP[-7]$ are exchanged, as well as the content of the pair $invP[8]$ and $invP[-1]$.

3.2 General Case

Up to this point, we assumed that the gene system is based on a set of integers, but genomes are usually represented by sequences of gene names (strings) in genome rearrangement data inputs. These sequences of strings must be converted to a proper representation (sequences of integer numbers) in order to be used efficiently in the arrays P, $invP$, and the list Q. To clarify the discussion, we use the following notation: genes are denoted by the letters x, y, z, w, while the indexes associated to genes are denoted by i, j, k, l. Suppose that we represent the gene system in the input of the problem as an array of genes L containing all the gene names in the gene system (E, Γ). We assume that the array L is a concatenation of the sets E_+ and E_- in this order in a oriented partition E_+ and E_- of (E, Γ) such that $L[|E_+| + i] = \Gamma L[i]$ for $1 \leq i \leq |E_+|$. We include a hash table H to store the indexes of the genes in L. Each gene name $L[i] = x$ is a key and the data record for the key x is the index $H[x].index = i$. The indexes of L are used in the same role as the genes for the arrays P and $invP$ are used in the special case that E is a set of integers.

If the set of gene names is the same for any input data set then a perfect hash table [3] implementation allows one to take a gene name as key and convert it to its index in the arrays in $O(|s|)$ running time where $|s|$ is the length of the gene name and to access the memory at most $O(1)$ times. On the other hand, if the set of genes is different for each data set then we use a chaining hash table and we choose an *addictive hash function* [13, 3]:

$$f(s) = \sum_{i=1}^{|s|} char(s_i)|A|^{i-1} \pmod m$$

where s is the gene name, the function $char(s_i)$ returns the ASCII code for the character s_i in the alphabet A, the length of the alphabet is $|A|$, and the number m is the number of entries in the hash table. We choose m as a prime number greater than the total number of genes in the gene system. Function $f(s)$ can be obtained in linear time on $|s|$ using Horner's rule for fast polynomial evaluation. We choose this hash function because it is simple and it seems to perform well for keys of small size, as it is the case of gene names. However, faster implementations and functions with better behavior should be used for data sets involving very large genomes with long gene names that are very similar to each other [11, 10]. We assume that a adequately designed hash table retrieves the data for a key s in $h = O(|s|)$ time.

For the general case the initialization of arrays P and S are slightly modified. We make the assumption that the genomes π and σ in the input of the procedure initialize are represented by a list of chromosomes and each chromosome is a list of genes in the circular order that they appear in both strands of the chromosome. For each gene name x in the list of chromosomes of π, the index i of x and the index j of πx in H are retrieved and j is assigned to $P[i]$. The same procedure is used for the initialization of S. Fig. 7 illustrates the procedure of initialization of P.

The query for an element $x \in E$ in genome π (or σ) over E consists of finding πx (or σx) by retrieving the index i of the gene x in the hash table H, and returning $L[P[i]]$ (or $L[S[i]]$). The query operation takes $O(h)$ running time to be performed.

```
initialize(Genome π, Table T, Gene List L)
1.    for each gene x in π do
2.        index = H[L[i]].index;
3.        P[index] = H[πL[i]].index;
4.    return array P
```

Figure 7: Procedure used to initialize arrays P and S.

The update of P and $invP$, by using the procedure *exchange* for instance, and the update of the list Q both are not modified by the inclusion of gene names to the input since all the operations that update P, $invP$, and Q do not require changes in the indexes in H. Each update operation on P, $invP$, and Q takes $O(1)$ time.

Fig. 8 illustrates the algorithm for finding a sequence of 2-breaks that transforms the genome π into σ both in the gene system (E, Γ).

Lemma 4. *Given π, σ, and Γ over E, the FFSRsort algorithm returns a sequence of good events for (π, σ) with minimum distance $d(\pi, \sigma)$ that transforms genome π into σ in $O(nh)$ running time, where $n = |E|$.*

Proof. We show that the algorithm FFSRSort is correct by defining the following loop invariant over the parameter r: $\theta = \rho_r \dots \rho_1 \pi$ and rearrangement event ρ_i is a good event for $(\rho_{i-1} \dots \rho_1 \pi, \sigma)$ applicable to $\rho_{i-1} \dots \rho_1 \pi$, for $1 \leq i \leq r$.

For $r = 0$, before the main loop in the line 7, we have $\theta = \pi$ and the invariant is trivially valid.

Suppose that the invariant is valid for $r = k$, that is, we have $\theta = \rho_k \dots \rho_1 \pi$ and ρ_i is a good event for $(\rho_{i-1} \dots \rho_1 \pi, \sigma)$ applicable to $\rho_{i-1} \dots \rho_1 \pi$ for $1 \leq i \leq k$. In the next iteration of the loop in line 10 we set $\rho_{k+1} = (x \ \sigma\theta^{-1}x)(\theta\Gamma\sigma\theta^{-1}x \ \theta\Gamma x)$ since $x = L[i]$ and $\sigma\theta^{-1}x = L[j]$. The rearrangement event ρ_{k+1} is a good event for (θ, σ) by Lemma 3. Therefore the invariant remains valid before the next iteration of the loop in line 7.

If $\theta = \rho_r \dots \rho_1 \pi$ and $r = d(\pi, \sigma)$ then $\theta = \sigma$, i.e. $\sigma\theta^{-1} = \iota$. If $\sigma\theta^{-1} = \iota$ then the list Q is empty because it does not contain trivial pairs. Therefore the condition in the line 7 is false and the algorithm executes the code in the line 26. At this point in the execution we have $\theta = \rho_r \dots \rho_1 \pi$ such that each ρ_i is a good event for $(\rho_{i-1} \dots \rho_1 \pi, \sigma)$ applicable to $\rho_{i-1} \dots \rho_1 \pi$, for $1 \leq i \leq r$ since the loop invariant is valid. Therefore $\sigma = \rho_r \dots \rho_1 \pi$ and $\rho_1, \ \dots, \ \rho_r$ is a sequence of good events, such that ρ_i is a good event for $(\rho_{i-1} \dots \rho_1 \pi, \sigma)$ and it is applicable to $\rho_{i-1} \dots \rho_1 \pi$, where $1 \leq i \leq r$, transforming genome π into genome σ.

Now we discuss how long it takes to execute the algorithm. The hash table H takes $O(nh)$ to be constructed, since each gene is read from the input, its name taken as a key in the hash function, and its index stored in a new node in H. The initialization of the arrays

Algorithm FFSRSort(Genome π, Genome σ, Genes L)

1. $r = 0$;
2. *for each* $i \in \{j \mid 1 \leq |j| \leq |E_+|\}$ *do* $T[L[i]].index = i$;
3. $P =$ initialize(π, T, L);
4. *for each* $i \in \{j \mid 1 \leq |j| \leq |E_+|\}$ *do* $invP[P[i]] = i$;
5. $S =$ initialize(σ, T, L);
6. initializePairsList(Q,S,$invP$);
7. *while* ($notEmpty(Q)$) *do* {
8. $r++$;
9. Take i, $j = S[invP[i]]$ in Q;
10. Make $\rho_r = (L[i]\ L[j])(L[P[-j]]\ L[P[-i]])$;
11. $remove(Q, j)$;
12. $remove(Q, P[-i])$;
13. $T[f(L[j])].node = NIL$;
14. $T[f(L[P[-i]])].node = NIL$;
15. *if* (trivialPair(Q,i)) {
16. $pair = T[i].pair$;
17. $remove(Q, i)$;
18. $remove(Q, P[-j])$;
19. $delete(pair)$;
20. }
21. $exchange(P[invP[i]], P[invP[j]])$;
22. $exchange(P[-j], P[-i])$;
23. $exchange(invP[i], invP[j])$;
24. $exchange(invP[P[-j]], invP[P[-i]])$;
25. }
26. *return* $\rho_1, \ldots, \rho_r$

Figure 8: The **FFSRSort** procedure. The algorithm returns the sequence of fissions, fusions, and signed reversals that transforms the genome π into the genome σ.

P and S takes $O(nh)$ running time since for each element x in E, in the order that they appear in the mappings of π and σ, must be assigned its πx and σx what involves accessing H and retrieving the index of each πx and σx respectively. Since we make the assumption that π and σ are represented in the input by a list of mappings $(x, \pi x)$ (or $(x, \sigma x)$) then it takes constant time to obtain πx (or σx) given x. The array $invP$ can be initialized by assigning $invP[P[i]] = i$ for $i \in \{x \leq n \mid 1 \leq |x| \leq |E_+|\}$, so it takes $O(n)$ running time.

Constructing the list Q takes $O(n)$ time because finding $S[invP[i]]$ for $i \in \{x \leq n \mid 1 \leq |x| \leq |E_+|\}$ involves a query in $invP$ and S that takes $O(1)$ and the assignment of the pointers in the entry $T[i]$ that takes $O(1)$.

In the worst case, just one element in each strand of the genome will be placed in its proper position per iteration of the loop in line 7, i.e. the block *while* will be executed $O(n)$ times. For each step in the *while* loop it is verified whether genomes θ and σ are the same in line 7. This verification can be accomplished in $O(1)$ running time by checking whether the list Q is empty. If Q is empty then $\theta = \sigma$ since $\sigma\pi^{-1} = \iota$. Finding a fission, fusion, or signed reversal that is a good event for (θ, σ) is performed in $O(1)$ time by choosing the elements i and j, corresponding to x and $\sigma\theta^{-1}x$, where i is the first element in the first pair of Q. The rearrangement event $(x\ \sigma\theta^{-1}x)(\theta\Gamma\sigma\theta^{-1}x\ \theta\Gamma x)$ is a good event for (θ, σ) where θ is the genome represented by the array P in the current iteration of the loop. The update process of the data structures can be achieved in $O(1)$ running time since the exchange operations on the arrays P and $invP$ take constant time. Removing j and $P[-i]$ from Q takes $O(1)$ since it takes $O(1)$ to remove the nodes in Q that contain them the pointers to those two nodes in $T[j]$ and $T[P[-i]]$. Therefore the total time complexity is $O(nh)$. □

Theorem 1. *Given genomes π, σ, and the function Γ, all over E, we have*

$$d(\pi, \sigma) = \frac{\|\sigma\pi^{-1}\|}{2}.$$

Proof. Given genomes π and σ over E, Lemma 4 guarantees the existence of a sequence of rearrangement events $\rho_1, \ldots, \rho_k$ such that $\rho_k \ldots \rho_1\pi = \sigma$ and ρ_i is applicable to $\rho_{i-1} \ldots \rho_1\pi$ and ρ_i is a good event for $(\rho_{i-1} \ldots \rho_1\pi, \sigma)$ for $1 \leq i \leq k$. In addition, by Lemma 2, we have $d(\pi, \sigma) \geq \|\sigma\pi^{-1}\|/2$ and $k = \|\sigma\pi^{-1}\|/2$. Therefore $d(\pi, \sigma) = \|\sigma\pi^{-1}\|/2$. □

Given the genomes π and σ over E, Theorem 1 offers a simple formula for the rearrangement distance $d(\pi, \sigma)$. We can obtain $d(\pi, \sigma)$ in $O(nh)$ running time, by finding $\sigma\pi^{-1}$ and its norm. It is the same asymptotic running time complexity for finding a sequence of good events that transforms the genome π into σ, although an implementation of the list Q construction that avoids to access the hash table would have lower constants.

4 Conclusion

Genome rearrangement analysis involving signed reversals, fissions, and fusions may be an important technique for genome comparison. We present a formalism for modeling genomes and rearrangement events that could be implemented by fast and simple data structures. We use such data structures to design an algorithm that finds a sequence of signed reversals,

fissions, and fusions whose length is the genomic distance. The algorithm properly handle multichromosomal genomes and it performs each of those rearrangement events in constant time. The total running time of an algorithm is $O(nh)$. Our algorithm may be the first one to find a sequence of rearrangement events that solves a rearrangement problem in linear time. Future research will involve the application of the same data structures and the algebraic formalism to another rearrangement problems, such as the problem including block-interchanges, for instance.

Acknowledgment

This work was supported by FAPESP (03/0731-3).

References

[1] V. Bafna and P. A. Pevzner, "Genome rearrangements and sorting by reversals," in *Proc. of the 34th Annual Symposium on Foundations of Computer Science (FOCS'93)*, IEEE, Ed. Palo Alto, USA: IEEE Computer Society Press, November 1993, pp. 148–157.

[2] A. Bergeron, "A very elementary presentation of the Hannenhalli-Pevzner theory," *Discrete Applied Mathematics*, vol. 146, no. 2, pp. 134–135, 2005.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms - Second Edition*. McGraw-Hill, 2001.

[4] Z. Dias and J. Meidanis, "Genome rearrangements distance by fusion, fission, and transposition is easy," in *Proc. of the String Processing and Information Retrieval (SPIRE'2001)*. Laguna de San Rafael, Chile: IEEE Computer Society, November 2001, pp. 250–253.

[5] P. Feijao and J. Meidanis, "Extending the algebraic formalism for genome rearrangements to include linear chromosomes", *Computational Biology and Bioinformatics*, IEEE/ACM Transactions on vol. 10, no. 4, pp. 819–831, 2013.

[6] S. Hannenhalli and P. A. Pevzner, "Transforming men into mice (polynomial algorithm for genomic distance problem)," in *Proc. of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*. Los Alamitos, USA: IEEE Computer Society Press, Oct. 1995, pp. 581–592.

[7] S. Hannenhalli and P. A. Pevzner, "Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals," *Journal of the ACM*, vol. 46, no. 1, pp. 1–27, Jan. 1999.

[8] T. Hartman and R. Shamir, "A simpler and faster 1.5-approximation algorithm for sorting by transpositions," in *Proceedings of CPM'03*, 2003, pp. 156 – 169, extended version.

[9] Y.-L. Huang and C. L. Lu, "Sorting by reversals, generalized transpositions, and translocations using permutation groups", *Journal of Computational Biology*, vol. 17, no. 5, pp. 685–705, 2010.

[10] P. Hsieh, "Hash functions," 2004, http://www.azillionmonkeys.com/qed/hash.html.

[11] R. J. J. Jr., "Hash functions for hash table lookup," 1997, http://burtleburtle.net/bob/hash/evahash.html.

[12] H. Kaplan, R. Shamir, and R. E. Tarjan, "Faster and simpler algorithm for sorting signed permutations by reversals," *SIAM Journal on Computing*, vol. 29, no. 3, pp. 880–892, Jan. 2000.

[13] D. E. Knuth, *The Art of Computer Programming: Sorting and Searching*. Addison-Wesley, 1973, vol. 3.

[14] J. Meidanis and Z. Dias, "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*, D. Sankoff and J. H. Nadeau, Eds. Kluwer Academic Publishers, Nov. 2000, pp. 213–223.

[15] J. Meidanis, M. E. M. T. Walter, and Z. Dias, "Reversal distance of signed circular chromosomes," Institute of Computing - University of Campinas, Tech. Rep. IC-00-23, Dec. 2000.

[16] C. Mira and J. Meidanis, "Sorting by Block-Interchanges and Signed Reversals". In *Proc. of the 14th International Conference on Information Technology: New Generations ITNG*, pp. 670—676. IEEE Computer Society, Las Vegas, USA, 2007.

[17] P. A. Pevzner and M. S. Waterman, "Open combinatorial problems in computational molecular biology," in *Proc. of the Third Israel Symposium on Theory of Computing and Systems*. IEEE Computer Society Press, 1995, pp. 158–163.

[18] S. Yancopoulos, O. Attie, and R. Friedberg, "Efficient sorting of genomic permutations by translocation, inversion and block interchange," *Bioinformatics*, vol. 21, no. 16, pp. 3340–3346, June 2005.