# Extending the Algebraic Formalism for Genome Rearrangements to Include Linear Chromosomes

Pedro Feijao[1] and Joao Meidanis[1,2]

[1] Institute of Computing - University of Campinas
ra932015@ic.unicamp.br
[2] Scylla Bioinformatics
meidanis@scylla.com.br

**Abstract.** Algebraic rearrangement theory, as introduced by Meidanis and Dias, focuses on representing the order in which genes appear in chromosomes, and applies to circular chromosomes only. By shifting our attention to genome adjacencies, we are able to extend this theory to linear chromosomes in a very natural way, and extend the distance formula to the general multichromosomal case, with both linear and circular chromosomes. The resulting distance, which we call algebraic distance here, is very similiar to, but not quite the same as, DCJ distance. We present linear time algorithms to compute it and to sort genomes. We also show how to compute the algebraic distance from the adjacency graph. Some results on more general $k$-break distances are given, with algebraic distance being 2-break distance under our interpretation.

**Keywords:** genome rearrangement

## 1 Introduction

The genome rearrangement problem can be seen as follows. We are given two genomes $\pi$ and $\sigma$ and need to return a most parsimonious sequence of rearrangement operations that transforms $\pi$ into $\sigma$. Important issues to consider are the kinds of operations permitted, as well as the notion of parsimony used.

Typically, allowed operations include reversals, transpositions, translocations, fusions, fissions, and, at times, block interchanges. These are what we will term *classical operations*. Some of them have been treated in isolation in the literature; in other articles, two or more operation types were considered. In general, parsimony is defined as using the fewest operations. Sometimes, however, especially when more than one operation kind is involved, different weights are assigned to each kind, and a solution of minimum total weight is sought. Weight choice is still a matter of debate.

As research on this topic progresses, we see models including more and more operation types. There is a balance between including all biologically sound, relevant operations, and the possibility of solving the resulting combinatorial problem efficiently. Yancopoulos et al. [1] introduced a new operation, called *double-cut-and-join* (DCJ), that models all classical operations, with weight 2 for

transpositions and block interchanges, and weight 1 for reversals, translocations, fusions, and fissions. This weight assigment is not unlike many others used in the literature [2–4].

In terms of formal models, the first papers on genome rearrangements dealt with unichromosomal genomes, and represented chromosomes as lists of genes, possibly signed to indicate orientation. This representation is not unique, though, because the reverse complement of a gene list represents the same chromosome. Besides, a circular chromosome can be represented by several lists, depending on where one starts the list.

In 2000, Meidanis and Dias [5] introduced a way of looking at chromosomes as bijections from genes to genes, that is, permutations over a certain gene set. This algebraic formalism is interesting because many results can be stated and proved in the realm of permutation group theory. The approach applies directly to circular chromosomes; for linear chromosomes, the usual tactic is to circularize them by means of "dummy" elements, which are removed at the end. Many papers were written on this subject. Huang and Lu [6] present a relatively recent account.

Another way of uniquely specifying a multichromosomal genome, with both linear and circular chromosomes, is by listing the adjacencies between gene extremities. Bergeron, Mixtacki, and Stoye [7] used this representation and a structure called the adjacency graph, to unify the study of rearrangement problems. Feijao and Meidanis [8] also used it to define a new operation, *single-cut-or-join* (SCJ), which gives rise to a distance related to the number of breakpoints between two genomes, and for which hard problems, such as finding a median genome, have straightforward, efficient solutions.

In this paper, we suggest a new way of modeling genomes which can be seen as a mixture of algebraic rearrangement theory with the adjacency formalism. As a result, linear chromosomes are modeled directly, without the need of "dummy" elements. The new theory inherits many results from the algebraic theory, including the elegant distance formula $d = \frac{\|\sigma\pi^{-1}\|_2}{2}$. Sorting by DCJ operations can be achieved in linear time [7], provided one does not insist in knowing the *type* of each operation, just the operations themselves. Contrast this with the results from Mira and Meidanis [3], and Huang and Lu [6], who output typed operations but take quadratic time.

The weights assigned to the operations are slightly different from the DCJ weights. As a result, the two distances are close but not identical. This opens up the possibility of solving hard problems, such as finding medians, or optimally reconstructing ancestors in a given phylogenetic tree, more efficiently.

## 2 Algebraic rearrangement theory

Meidanis and Dias [5] introduced a model where permutation group theory was used to model genome rearrangement problems, the *Algebraic rearrangement theory*. In its original form, it is limited to circular chromosomes only, and it was used to solve several problems with different rearrangement operations [2–4].

We will present some basic concepts of this theory and then introduce an extended algebraic theory, which we call *adjacency algebraic theory*, that will allow the modeling of linear chromosomes.

### 2.1 Basic Concepts

Given a set $E$, a *permutation* $\alpha$ is a map from $E$ onto itself, that is, $\alpha : E \to E$. Permutations are represented with each element followed by its image. For instance, with $E = \{a, b, c\}$, $\alpha = (a\ b\ c)$ is the permutation where $a$ is mapped to $b$, which is mapped to $c$, which in turn is mapped back to $a$. This representation is not unique; $(b\ c\ a)$ and $(c\ a\ b)$ are equivalent. Figure 1(a) gives a graphical representation of this permutation.

Permutations are composed of one or more *cycles*. For instance, the permutation $\alpha = (a\ b\ c)(d\ e)(f)$ has three cycles. A cycle with $k$ elements is called a *$k$-cycle*. An 1-cycle represents a fixed element in the permutation and is usually omitted.

The *support* of a permutation is the set of its non-fixed elements. In the previous example, $Supp(\alpha) = \{a, b, c, d, e\}$.

The *product* or *composition* of two permutations $\alpha$, $\beta$ is denoted by $\alpha\beta$. The product $\alpha\beta$ is defined as $\alpha\beta(x) = \alpha(\beta(x))$ for $x \in E$. For instance, with $E = \{a, b, c, d, e, f\}$, $\alpha = (b\ d\ e)$ and $\beta = (c\ a\ e\ b\ f\ d)$, we have $\alpha\beta = (c\ a\ b\ f\ e\ d)$.

In general $\alpha\beta \neq \beta\alpha$, but when $\alpha$ and $\beta$ are disjoint cycles, that is, don't have any element in common, they commute: $\alpha\beta = \beta\alpha$. Every permutation can be written in an unique way as a product of disjoint cycles; this is called the *cycle decomposition* of a permutation.

The *identity permutation*, which maps every element into itself, will be denoted by $\mathbf{1}$. Every permutation $\alpha$ has an *inverse* $\alpha^{-1}$ such that $\alpha\alpha^{-1} = \alpha^{-1}\alpha = \mathbf{1}$. For a cycle, the inverse is obtained by reverting the order of its elements: $(c\ b\ a)$ is the inverse of $(a\ b\ c)$.

The *conjugation* of $\beta$ by $\alpha$, denoted by $\alpha \cdot \beta$, is the permutation $\alpha\beta\alpha^{-1}$. This results in a permutation with the same structure as $\beta$, but with $\alpha$ applied to each element. For instance, if $\beta = (b_1\ b_2\ \ldots\ b_n)$ then $\alpha \cdot \beta = (\alpha b_1\ \alpha b_2\ \ldots\ \alpha b_n)$, where $\alpha b_i$ is a simpler notation for $\alpha(b_i)$.

A *$k$-cycle decomposition* of a permutation $\alpha$ is a representation of $\alpha$ as a product of $k$-cycles, not necessarily disjoint. All permutations have a 2-cycle decomposition. The *$k$-norm* of a permutation $\alpha$, denoted by $\|\alpha\|_k$, is the minimum number of cycles in a $k$-cycle decomposition of $\alpha$. The *norm* of a permutation is defined as the 2-norm, and the subscript can be omitted, that is, $\|\alpha\| \equiv \|\alpha\|_2$.

### 2.2 Modeling Genomes with Algebraic Theory

We base our definitions on the pioneering work by Meidanis and Dias [5], except that their $\gamma$ is written as $\Gamma$ here, following more recent literature [3]. Let $E_n = \{-1, +1, -2, +2, \ldots, -n, +n\}$, where $n$ is the number of genes, as the base set to model genomes as permutations, representing all genes in both orientations.

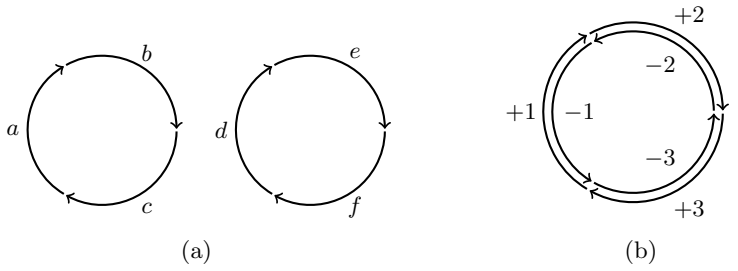


Fig. 1: (a) Graphic representation of the permutation $(a\ b\ c)(d\ e\ f)$, composed by two 3-cycles; (b) a circular genome represented by the permutation $\pi = (+1\ +2\ +3)(-3\ -2\ -1)$, with two 3-cycles, one for each strand

Let $\Gamma$ be the permutation that maps each gene into its reverse complement, that is, $\Gamma = (-1\ +1)(-2\ +2)\dots(-n\ +n)$. A *chromosome* is the product of two cycles $\alpha$ and $\Gamma \cdot \alpha^{-1}$, representing the strands of the chromosome. A *genome* is the product of disjoint chromosomes. A necessary and sufficient condition for a permutation $\pi$ to represent a valid genome is: (1) $\Gamma\pi\Gamma = \pi^{-1}$ and (2) no strand in $\pi$ contains both $-i$ and $+i$ for any gene $i$.

For instance, the circular genome depicted in Figure 1 is modelled by $\pi = (+1\ +2\ +3)(-3\ -2\ -1)$. Notice that $(+1\ +2\ +3) = \Gamma \cdot (-3\ -2\ -1)^{-1}$, making it a valid product of cycles representing a chromosome, and also guaranteeing $\pi^{-1} = \Gamma\pi\Gamma$.

A *rearrangement operation* $\rho$ applicable to a genome $\pi$ is defined as a permutation $\rho$ for which $\pi' = \rho\pi$ is a valid genome. The *weight* of the operation $\rho$ is defined as $\|\rho\|/2$. With this definition, circular fusions and fissions are modelled with permutations formed with two 2-cycles, and have therefore weight 1, whereas transpositions are modelled with two 3-cycles, and block interchanges with four 2-cycles, both with a resulting weight of 2 [3]. This weight definition agrees with the DCJ weights for reversals and generalized transpositions, but differs from it on fusions and fissions, which are weighted 1 in DCJ [1] and 1/2 here.

With this background, we can formulate the *algebraic rearrangement problem* as finding permutations $\rho_1, \rho_2, \dots, \rho_n$ that minimally transform $\pi$ into $\sigma$, that is, $\rho_i \dots \rho_2\rho_1\pi$ is a valid genome for every $i$, $\rho_n \dots \rho_2\rho_1\pi = \sigma$, and $\sum_{i=1}^{n} \|\rho_i\|/2$ is minimum. This minimum value is called the *algebraic distance* between $\pi$ and $\sigma$, denoted by $d(\pi, \sigma)$.

It is not hard to see that $d(\pi, \sigma) = \|\sigma\pi^{-1}\|/2$: we can easily show that $\|\sigma\pi^{-1}\|/2$ is a lower bound for the distance with some algebraic manipulation of the definition. If $\rho_1, \rho_2, \dots, \rho_n$ minimally transform $\pi$ into $\sigma$, we have:

$$\frac{\|\sigma\pi^{-1}\|}{2} = \frac{\|\rho_n \dots \rho_2\rho_1\|}{2} \leq \frac{\sum_{i=1}^{n} \|\rho_i\|}{2} = d(\pi, \sigma) \tag{1}$$

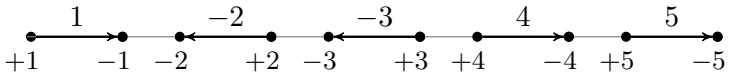


Fig. 2: A genome composed of just a linear chromosome, with gene extremities labeled according to the set theoretical genome representation, also used in our adjacency algebraic theory.

On the other hand, $\sigma\pi^{-1}$ itself can be considered a (possibly very heavy) rearrangement operation, because $(\sigma\pi^{-1})\pi = \sigma$ is a valid genome, and its weight equals the lower bound, showing that this is in fact the distance value.

Classical definition restricts the available operations to small weight permutations, such as just fusions, fissions, reversals, etc. We will show in Section 3.2 that one can achieve the same distance using only operations of weight 1 or less.

Although several results in genome rearrangement theory have been achieved using this model, in its original form it can only model circular chromosomes. We will expand it to allow linear chromosomes as well by introducing an additional genome representation scheme, which will expand the theory but at the same time maintain most of its important results.

### 2.3 Adjacency Algebraic Theory

The original algebraic formulation focuses on representing the order in which genes appear in chromosomes, and the "circular" nature of permutations forces it to be applied to circular chromosomes only. In our proposed formulation, we shift our attention to genome adjacencies, adequately extending this theory to the general multichromosomal case, with both linear and circular chromosomes, while keeping most of the properties of the original formulation. We call it the *adjacency algebraic theory*.

This formulation is similar to the set representation of a genome [7, 8]. In this representation, each gene $a$ has two *extremities*, called *tail* and *head*, respectively denoted by $a_t$ and $a_h$, or alternatively using signs, where $-a = a_h$ and $+a = a_t$. An *adjacency* is an unordered pair of extremities indicating a linkage between two consecutive genes in a chromosome. An extremity not adjacent to any other extremity in a genome is called a *telomere*. A genome is represented as a set of adjacencies and telomeres (possibly omitted, when the gene set is given) where each extremity appears at most once. For instance, the genome in Fig. 2 is represented by $\{\{+1\}, \{-1, -2\}, \{+2, -3\}, \{+3, +4\}, \{-4, +5\}, \{-5\}\}$ or just by $\{\{-1, -2\}, \{+2, -3\}, \{+3, +4\}, \{-4, +5\}\}$.

In the adjacency algebraic theory, genomes are also representated by permutations, but a *genome* is a product of 2-cycles, where each 2-cycle corresponds to an adjacency. Therefore, the genome in Fig. 2 is represented as $\pi' = (-1\ -2)(+2\ -3)(+3\ +4)(-4\ +5)$. Note that in this representation telomeres can be safely omitted, since they are 1-cycles. With this formulation, linear chromosomes can be represented.

The first property of this new representation is that there is a direct relationship with the original one. Specifically, if $\pi$ is a permutation representing a circular genome in the original algebraic formulation, then $\pi' = \pi\Gamma$ represents the same genome in the adjacency theory. For instance, for the genome in Fig. 1, we have $\pi = (+1\ +2\ +3)(-3\ -2\ -1)$ and $\pi' = \pi\Gamma = (-1\ +2)(-2\ +3)(-3\ +1)$.

Another property is that the rearrangement events are represented by the same permutations in both formulations. If $\pi$ and $\sigma$ are genomes in the original formulation and $\rho\pi = \sigma$, then by multiplying by $\Gamma$ on the right we easily get $\rho\pi' = \sigma'$, where $\pi'$ and $\sigma'$ represent the same genomes in the adjacency formulation. Also, $\sigma'\pi'^{-1} = \sigma\Gamma(\pi\Gamma)^{-1} = \sigma\Gamma\Gamma^{-1}\pi^{-1} = \sigma\pi^{-1}$. So, both the rearrangement operations and the all-important permutation $\sigma\pi^{-1}$ remain the same in the adjacency theory, meaning that we have many results (all related to circular genomes) already demonstrated. Therefore, we will have to develop new results in order to extend the algebraic theory for linear chromosomes, specifically to treat the cases where telomeres appear in the permutations, as we will see in the following sections.

## 3 Sorting by algebraic operations

A permutation $\rho$ is called a *sorting operation* from $\pi$ to $\sigma$ if $\rho\pi$ is a valid genome and $d(\rho\pi, \sigma) = d(\pi, \sigma) - \|\rho\|/2$, that is, applying $\rho$ in $\pi$ decreases the distance between $\pi$ and $\sigma$ by the weight of operation $\rho$, that we already defined as being half the norm. But we see that $d(\rho\pi, \sigma) = d(\pi, \sigma) - \|\rho\|/2 \Rightarrow \|\sigma(\rho\pi)^{-1}\|/2 = \|\sigma\pi^{-1}\|/2 - \|\rho\|/2 \Rightarrow \|(\sigma\pi^{-1})\rho^{-1}\| = \|\sigma\pi^{-1}\| - \|\rho\|$. Therefore, to say that $\rho$ decreases the distance by its weight is the same as $\rho|\sigma\pi^{-1}$, that is, $\rho$ divides $\sigma\pi^{-1}$. Then, a sorting operation from $\pi$ to $\sigma$ is a valid operation on $\pi$ that divides $\sigma\pi^{-1}$. Some results on valid operations and divisibility are given in the following lemmas:

**Lemma 1 (Valid operations).** *Given a genome $\pi$, a permutation $\rho$ is a valid operation on $\pi$, that is, $\rho\pi$ is a valid genome, if and only if $\pi \cdot \rho = \rho^{-1}$.*

*Proof.* A permutation $\pi$ is a valid genome if $\pi^2 = \mathbf{1}$. If $\rho\pi$ is valid, then $(\rho\pi)^2 = \mathbf{1}$ and $(\rho\pi)^2 = \rho\pi\rho\pi = \rho(\pi \cdot \rho) = \mathbf{1}$, since $\pi = \pi^{-1}$. Then $\pi \cdot \rho$ is the inverse of $\rho$. On the other hand, if $\pi \cdot \rho = \rho^{-1}$, then $(\rho\pi)^2 = \rho\pi\rho\pi = \rho(\pi \cdot \rho) = \rho\rho^{-1} = \mathbf{1}$. □

**Corollary 1.** *Any permutation $\rho$ that can be written as $\rho = \mu(\pi \cdot \mu^{-1})$ is a valid operation in $\pi$.*

*Proof.* It is easy to see that $\pi \cdot \rho = \rho^{-1}$, then by Lemma 1 $\rho$ is valid on $\pi$. □

This corollary is important because permutations in the form $\rho = \alpha(\pi \cdot \alpha^{-1})$ will be the basis of our sorting operations, as we will see when we study the permutation $\sigma\pi^{-1}$ below. Now we will show an important result about permutation divisibility, that applying the reverse conjugation mantains divisibility:

**Lemma 2.** *Given a genome $\pi$ and a permutation $\alpha$, for any permutation $\mu$ where $\mu|\alpha$ we have $\pi \cdot \mu^{-1}|\pi \cdot \alpha^{-1}$.*

*Proof.* Since $\mu|\alpha$, we have $\|\alpha\mu^{-1}\| = \|\alpha\| - \|\mu\|$. Then:

$$\|(\pi \cdot \alpha^{-1})(\pi \cdot \mu^{-1})^{-1}\| = \|\pi \cdot (\alpha^{-1}\mu)\| = \|\alpha^{-1}\mu\| = \|\alpha\mu^{-1}\| = \|\alpha\| - \|\mu\|$$

Since the operations of conjugation and inverse do not change the norm of a permutation, we have $\|\alpha\| = \|\pi \cdot \alpha^{-1}\|$ and $\|\mu\| = \|\pi \cdot \mu^{-1}\|$, and then

$$\|(\pi \cdot \alpha^{-1})(\pi \cdot \mu^{-1})^{-1}\| = \|\pi \cdot \alpha^{-1}\| - \|\pi \cdot \mu^{-1}\|$$

the exact definition of $\pi \cdot \mu^{-1}|\pi \cdot \alpha^{-1}$. □

Using Lemma 2 we get to the following lemma, where the proof was left out for space reasons.

**Lemma 3.** *Given a genome $\pi$ and a permutation $\tau = \alpha(\pi \cdot \alpha^{-1})$ where $\|\tau\| = \|\alpha\| + \|\pi \cdot \alpha^{-1}\|$, for any permutation $\mu$ where $\mu|\alpha$ we have that the permutation $\rho = \mu(\pi \cdot \mu^{-1})$ divides $\tau$, that is, $\rho|\tau$, and also $\rho$ is a valid operation on $\pi$.*

It should be noted that finding a cycle $\mu = (e_1 \ \dots \ e_k)$ that divides a cycle $\alpha$ is easy, if we choose $e_1, \dots, e_k$ as a subset of elements of $\alpha$ such that they also appear in $\alpha$ in the same order. For instance, if $\alpha = (1\ 2\ 3\ 4\ 5)$, then $\mu = (1\ 3\ 4)$ divides $\alpha$, but $\mu' = (1\ 4\ 3)$ does not. A formal proof of this was shown by Huang and Lu [6, Lemma 2.7].

A last lemma exposes the relationship of these $\mu(\pi \cdot \mu^{-1})$ operations with the $k$-break operation, introduced by Alekseyev and Pevzner [9]. A $k$-break is an operation that cuts $k$ adjacencies in $\pi$ and then joins $k$ new ones within the same extremities.

**Lemma 4.** *A permutation $\rho = \mu(\pi \cdot \mu^{-1})$ where $\mu$ and $\pi \cdot \mu^{-1}$ are disjoint and $||\mu|| = k - 1$, is a $k$-break operation on $\pi$.*

*Proof.* Let $\mu = (e_1\ e_2\ \cdots\ e_k)$, so $\|\mu\| = k - 1$, and let $\rho = \mu(\pi \cdot \mu^{-1})$. If $\mu$ and $\pi \cdot \mu^{-1}$ are disjoint, $\pi$ has the adjacencies $(e_1\ \pi e_1)\dots(e_k\ \pi e_k)$ and applying $\rho$ we can see that $\rho\pi$ will have the adjacencies $(e_1\ \pi e_k)(e_2\ \pi e_1)\dots(e_k\ \pi e_{k-1})$. Therefore, $k$ adjacencies of $\pi$ were removed and changed by $k$ new ones, and $\rho$ is a $k$-break. □

In the next section we will use Lemma 3 to find sorting operations, that is, valid operations that divide $\sigma\pi^{-1}$.

### 3.1 Characterizing $\sigma\pi^{-1}$ and finding sorting operations

In this section we will use the *Adjacency Graph* between two genomes $\pi$ and $\sigma$, defined by Bergeron, Mixtacki, and Stoye [7]. In this graph, denoted as $AG(\pi, \sigma)$, the vertices are the adjacencies and telomeres of $\pi$ and $\sigma$, and for each $u \in \pi$ and $v \in \sigma$ there is an edge between $u$ and $v$ for each extremity that $u$ and $v$ have in common. We will show that every connected component in $AG(\pi, \sigma)$ has a direct relationship with a permutation in $\sigma\pi^{-1}$ and then determine sorting operations on these permutations.

**Cycles in $AG(\pi, \sigma)$** A cycle of size $n$ in $AG(\pi, \sigma)$ contains the following adjacencies as vertices, starting with an adjacency $(e_1, e_2)$ in $\pi$ and alternating vertices of $\pi$ and $\sigma$:

$$\underbrace{(e_1, e_2)}_{\pi}, \underbrace{(e_2, e_3)}_{\sigma}, \ldots, \underbrace{(e_{2k-1}, e_{2k})}_{\pi}, \underbrace{(e_{2k}, e_{2k+1})}_{\sigma}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\pi}, \underbrace{(e_n, e_1)}_{\sigma}$$

Therefore, adjacencies in $\pi$ will have the form $\pi_k = (e_{2k-1}, e_{2k})$ and in $\sigma$ the form $\sigma_k = (e_{2k}, e_{2k+1})$, for $k = 1, \ldots, n$ (assuming for simplicity that $e_{n+1} \equiv e_1$).

When we calculate the product $\sigma\pi^{-1}$ the adjacencies in the $AG(\pi, \sigma)$ cycle will be 2-cycles disjoint from the rest of the adjacencies in $\pi$ and $\sigma$. Therefore, it will be part of the cycle decomposition of $\sigma\pi^{-1}$. We can multiply the 2-cycles in this $AG(\pi, \sigma)$ cycle to obtain the restriction $\tau$ of $\sigma\pi^{-1}$ to the $AG(\pi, \sigma)$ cycle (notice that $\pi_i^{-1} = \pi_i$ for each $i$):

$$\begin{aligned}
\tau &= \sigma_1\sigma_2 \ldots \sigma_n\pi_1 \ldots \pi_n \\
\tau &= (e_2\ e_3) \ldots (e_{2k}\ e_{2k+1}) \ldots (e_n\ e_1)(e_1\ e_2) \ldots (e_{2k-1}\ e_{2k}) \ldots (e_{n-1}\ e_n) \\
\tau &= (e_n\ e_{n-2}\ \ldots\ e_4\ e_2)(e_1\ e_3\ \ldots\ e_{n-3}\ e_{n-1}) \\
\tau &= (e_n\ e_{n-2}\ \ldots\ e_4\ e_2)(\pi e_2\ \pi e_4\ \ldots\ \pi e_{n-2}\ \pi e_n) \\
\tau &= \alpha(\pi \cdot \alpha^{-1}),
\end{aligned}$$

where $\alpha = (e_n\ e_{n-2}\ \ldots\ e_4\ e_2)$. Therefore, a cycle of length $n$ in $AG(\pi, \sigma)$ corresponds to 2 $(n/2)$-cycles in $\sigma\pi^{-1}$, where one is the reversed $\pi$-conjugation of the other.

To extract sorting operations in this case, we see that $\tau$ satisfies Lemma 3, therefore any $\mu$ that divides $\alpha$ generates a sorting operation $\rho = \mu(\pi \cdot \mu^{-1})$, with weight $w = \|\rho\|/2 = \|\mu\|$.

**Odd Paths in $AG(\pi, \sigma)$** An odd path of size $n$ in $AG(\pi, \sigma)$, that is, a path with $n$ edges where $n$ is odd, starting with a telomere $e_1$ in $\pi$ and ending at a telomere $e_n$ in $\sigma$, has vertices

$$\underbrace{(e_1)}_{\pi}, \underbrace{(e_1, e_2)}_{\sigma}, \underbrace{(e_2, e_3)}_{\pi}, \ldots, \underbrace{(e_{2k-1}, e_{2k})}_{\pi}, \underbrace{(e_{2k}, e_{2k+1})}_{\sigma}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\pi}, \underbrace{(e_n)}_{\sigma}$$

Then, similarly to the previous case, computing the restriction $\tau$ of $\sigma\pi^{-1}$ to these adjacencies, we have

$$\tau = (\ e_n\ e_{n-2}\ \ldots\ e_3\ e_1\ e_2\ e_4\ \ldots\ e_{n-3}\ e_{n-1})$$

Therefore, an odd path in $AG(\pi, \sigma)$ corresponds to an $n$-cycle in $\sigma\pi^{-1}$. Notice that we can write this as a product of (nondisjoint) reversed $\pi$-conjugates:

$$\begin{aligned}
\tau &= (e_n\ e_{n-2}\ \ldots\ e_3\ e_1)(e_1\ e_2\ e_4\ \ldots\ e_{n-3}\ e_{n-1}) \\
\tau &= (e_n\ e_{n-2}\ \ldots\ e_3\ e_1)(\pi e_1\ \pi e_3\ \pi e_5\ \ldots\ \pi e_{n-2}\ \pi e_n) = \alpha(\pi \cdot \alpha^{-1})
\end{aligned}$$

where $\alpha = (e_n\ e_{n-2}\ \ldots\ e_3\ e_1)$, then $\tau$ satisfies Lemma 3, and with any $\mu$ such that $\mu|\alpha$ we derive a sorting operation $\rho = \mu(\pi \cdot \mu^{-1})$, with weight $w = \|\rho\|/2 = \|\mu\|$.

**Even Paths in $AG(\pi, \sigma)$** An even path of size $n$ in $AG(\pi, \sigma)$ will have both path extremities (telomeres) in the same genome. Then, we have two cases: both telomeres in $\pi$ or in $\sigma$.

*i) Both telomeres in $\sigma$.* If both telomeres are in $\sigma$, the vertices are of the form

$$\underbrace{(e_1)}_{\sigma}, \underbrace{(e_1, e_2)}_{\pi}, \underbrace{(e_2, e_3)}_{\sigma}, \ldots, \underbrace{(e_{2k-1}, e_{2k})}_{\pi}, \underbrace{(e_{2k}, e_{2k+1})}_{\sigma}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\pi}, \underbrace{(e_n)}_{\sigma}$$

Then, computing the restriction $\tau$ of $\sigma\pi^{-1}$, we have

$$\tau = (e_n\ e_{n-2}\ \ldots\ e_4\ e_2\ \pi e_2\ \pi e_4\ \ldots\ \pi e_{n-2}\ \pi e_n)$$

which is an $n$-cycle in $\sigma\pi^{-1}$, and with some manipulation we get to

$$\tau = (\pi e_2\ e_n)(e_n\ e_{n-2}\ \ldots\ e_4\ e_2)(\pi e_2\ \pi e_4\ \ldots\ \pi e_{n-2}\ \pi e_n)$$
$$\tau = (e_1\ e_n)\alpha(\pi \cdot \alpha^{-1})$$

that is, the product of a 2-cycle with $\alpha(\pi \cdot \alpha^{-1})$ permutation, where $\alpha = (e_n\ e_{n-2}\ \ldots\ e_4\ e_2)$.

*ii) Both telomeres in $\pi$.* If both telomeres are in $\pi$, the vertices are of the form

$$\underbrace{(e_1)}_{\pi}, \underbrace{(e_1, e_2)}_{\sigma}, \underbrace{(e_2, e_3)}_{\pi}, \ldots, \underbrace{(e_{2k-1}, e_{2k})}_{\sigma}, \underbrace{(e_{2k}, e_{2k+1})}_{\pi}, \ldots, \underbrace{(e_{n-1}, e_n)}_{\sigma}, \underbrace{(e_n)}_{\pi}$$

Then, the restriction $\tau$ of $\sigma\pi^{-1}$ will be

$$\tau = (e_{n-1}\ e_{n-3}\ \ldots\ e_3\ e_1\ \pi e_3\ \ldots\ \pi e_{n-3}\ \pi e_{n-1}\ e_n)$$

again an $n$-cycle in $\sigma\pi^{-1}$. With more algebrism with get to

$$\tau = (e_n\ e_{n-1})(e_{n-1}\ e_{n-3}\ \ldots\ e_3\ e_1\ \pi e_3\ \ldots\ \pi e_{n-3}\ \pi e_{n-1})$$
$$\tau = (e_n\ e_{n-1})\alpha(\pi \cdot \alpha^{-1})$$

and again we get to a product of a 2-cycle with a permutation in the form $\alpha(\pi \cdot \alpha^{-1})$.

In both even path cases, permutation $\alpha(\pi \cdot \alpha^{-1})$ is a part of $\tau$ and $\rho = \mu(\pi \cdot \mu^{-1})$, where $\mu|\alpha$ is a sorting operation with weight $w = \|\rho\|/2 = \|\mu\|$.

*iii) Special cases.* In both types of even paths there is one special case, specifically when $n = 2$, where permutation $\alpha$ becomes an 1-cycle and $\tau$ is reduced to only the 2-cycle, and in both cases $\tau = (e_1\ e_2)$. In the first case, both telomeres are in $\sigma$ and $\tau$ is a *cut* in $\pi$, splitting the adjacency of $(e_1\ e_2)$ of $\pi$ into two telomeres. It is easy to see that $\tau$ is a sorting operation. On the second case, both telomeres are in $\pi$, and $\tau$ is a *join* in $\pi$, joining telomeres $e_1$ and $e_2$ into one adjacency. Again, $\tau$ is a sorting operation. In both cases, this operation has weight 1/2, since it is formed by just one 2-cycle.

In this section we learned how to derive sorting operations from the permutation $\sigma\pi^{-1}$. Sorting operations are usually in the format $\rho = \mu(\pi \cdot \mu^{-1})$, where $\mu$ divides a cycle of $\sigma\pi^{-1}$, but there are also special cases of single 2-cycle operations like cuts and joins.

### 3.2 Algebraic sorting with 2-break (DCJ) operations

From the previous section we saw that we can always find a sequence $\rho_1, \ldots, \rho_n$ of sorting operations such that $\rho_n \ldots \rho_1 \pi = \sigma$, and $\sum_{i=1}^{n} \|\rho_i\|/2 = \|\sigma\pi^{-1}\|/2 = d(\pi, \sigma)$. But that leaves the following question: can we always find sorting operations $\rho_1, \ldots, \rho_n$ where $\rho_n \ldots \rho_1 \pi = \sigma$ and $\sum_{i=1}^{n} \|\rho_i\|/2 = d(\pi, \sigma)$, with the additional constraint that $\|\rho_i\|/2 \leq w$, for $i = 1, \ldots, n$, for any given $w$? It should be noted that when we choose different values of $w$, the distance does not change, since the weight of the rearrangement operations is always the same, but we change the *scenario* of the rearrangent sorting.

Of particular interest are operations of weight 1 or less, corresponding to 2-breaks, that we know from DCJ theory that correspond to all classic operations of reversal, translocation, fusion and fissons (generalized transpositions are also possible by applying two specific operations of weight 1). From Section 3.1 we see that it is always possible to find 2-break sorting operations. In all cases where the sorting operation is in the format $\rho = \mu(\pi \cdot \mu^{-1})$, if we choose $\mu$ as a 2-cycle, then the weight is $\|\rho\|/2 = \|\mu\| = 1$ and $\rho$ is a 2-break. There are also the special cases of cuts or joins, but in both cases the operation has weight 1/2. Therefore, using algebraic theory, it is always possible to find a rearrangement scenario using only operations of weight 1 or less, which means only classical operations are being used.

Also, it is not difficult to see that the operations found by the algorithm for sorting with DCJ operations by Bergeron, Mixtacki, and Stoye [7] are also sorting operations under the algebraic theory, which means that algebraic sorting by 2-breaks can be achieved in linear time.

### 3.3 Comparing the Algebraic with DCJ distance

To compare the algebraic and DCJ distances, we will use the graph $AG(\pi, \sigma)$ again. From Section 3.1, we know that any cycle of size $2n$ in $AG(\pi, \sigma)$ will correspond to two $n$-cycles in $\sigma\pi^{-1}$, and a path of size $n$ in $AG(\pi, \sigma)$ will become an $n$-cycle in $\sigma\pi^{-1}$. Since the norm of an $n$-cycle is $n-1$ and the algebraic weight of an operation is the norm divided by two, the cost of sorting a cycle of size $2n$ is $n-1$, and sorting a path of size $n$ costs $(n-1)/2$. Then, the algebraic distance can be computed as follows:

$$d(\pi, \sigma) = \sum_{k=1}^{n} (k-1) C_{2k} + \sum_{k=1}^{n} \frac{k-1}{2} P_k \tag{2}$$

where $C_{2k}$ is the number of cycles of size $2k$ and $P_k$ is the number of paths of size $k$ in $AG(\pi, \sigma)$. Also, we know that there are $4N$ extremities in the vertices of $AG(\pi, \sigma)$, where $N$ is the number of genes. Since each cycle of size $2k$ has $2k$ vertices, comprising of $4k$ extremities, and each path of size $k$ has $k+1$ vertices with a total of $2k$ extremities, we have

$$N = \sum_{k=1}^{n} k C_{2k} + \sum_{k=1}^{n} \frac{k}{2} P_k \tag{3}$$

Using (2) and (3) we have

$$d(\pi,\sigma) = N - \sum_{k=1}^{n} C_{2k} - \sum_{k=1}^{n} \frac{1}{2} P_k = N - (C + \frac{P}{2}) \tag{4}$$

where $C = \sum_{k=1}^{n} C_{2k}$ and $P = \sum_{k=1}^{n} P_k$ are respectively the number of cycles and paths in $AG(\pi,\sigma)$. Since $d_{DCJ} = N - (C + P_{odd}/2)$ [7], we have

$$d(\pi,\sigma) = d_{DCJ}(\pi,\sigma) - \frac{P_{even}}{2} \tag{5}$$

where $P_{odd}$ and $P_{even}$ denote the number of odd and even paths in $AG(\pi,\sigma)$, respectively.

This small difference is due to the fact that although most DCJ operations have the same weight in the algebraic theory, when sorting an even path at least one *cut* or *join* must be performed. Since these operations are modelled as permutations with a single 2-cycle, they have weight 1/2 under the algebraic theory, but weight 1 in the DCJ model, hence the difference in the distances.

Another difference is that the DCJ model allows operations that recombine two even paths into two odd paths [10]. We can see in the distance equations (4) and (5) that this kind of operation is indeed optimal in the DCJ model (that is, it reduces the distance by 1) but in the algebraic model the distance is not changed.

In addition to the distance formula, we also compared algebraic and DCJ distances with a scatterplot between randomly evolved genomes. Starting with a genome with 1000 genes and 5 chromosomes, we applied a random number of rearrangement operations and then measured the distance between the original and evolved genomes under both algebraic and DJC distances, resulting in the scatterplot shown in Figure 3.

## 4 Conclusions

The main accomplishment reported in this paper is an extension of the algebraic theory of genome rearrangements to include linear genomes in a quite natural way. The result is a new notion of genomic distance, which can be computed in linear time, and for which a list of small-weight, sorting operations can also be retrieved in linear time. This is better than the quadratic time obtained in previous papers.

In addition, the new distance yields values very close to the acclaimed DCJ distance. The difference is due to the weight given by cuts and joins in the two approaches: DCJ assigns them weight 1, whereas the algebraic theory assigns them weight 1/2. Since the distances are not exactly the same, the NP-hardness proofs for some important problems under DCJ, such as finding a median, do not immediately apply to the algebraic distance, leaving open the possibility of a polynomial algorithm for these problems. We remark that the recently introduced SCJ distance did just that with respect to breakpoints: allowed linear solutions for NP-hard problems under other, similar BP-like distances [8].

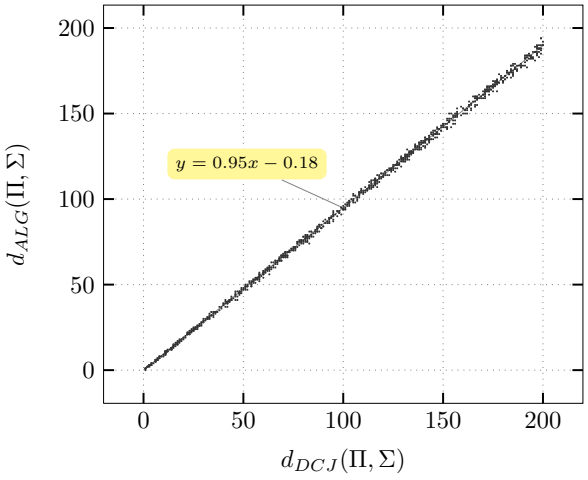


Fig. 3: Scatterplot comparing algebraic and DCJ distances between randomly evolved genomes. Linear regression on the points resulted in the equation $y = 0.95x - 0.18$

## References


1. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**(16) (2005) 3340–6
2. Lu, C.L., Huang, Y.L., Wang, T.C., Chiu, H.T.: Analysis of circular genome rearrangement by fusions, fissions and block-interchanges. BMC Bioinformatics **7** (2006) 295
3. Mira, C., Meidanis, J.: Sorting by block-interchanges and signed reversals. In: ITNG'07. (2007) 670–676
4. Dias, Z., Meidanis, J.: Genome rearrangements distance by fusion, fission, and transposition is easy. In: Proc. SPIRE 2001. (2001) 250–253
5. Meidanis, J., Dias, Z.: An Alternative Algebraic Formalism for Genome Rearrangements. In D. Sankoff and J. H. Nadeau, ed.: Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families. Kluwer Academic Publishers (2000) 213–223
6. Huang, Y.L., Lu, C.L.: Sorting by reversals, generalized transpositions, and translocations using permutation groups. Journal of Computational Biology **17**(5) (2010) 685–705
7. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. Algorithms in Bioinformatics **4175** (2006) 163–173
8. Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE IEEE/ACM Transactions on Computational Biology and Bioinformatics **8** (2011) 1318–1329
9. Alekseyev, M., Pevzner, P.: Multi-break rearrangements and chromosomal evolution. Theoretical Computer Science **395**(2-3) (2008) 193–202
10. Braga, M.D.V., Stoye, J.: The solution space of sorting by DCJ. Journal of Computational Biology **17**(9) (2010) 1145–65